NAVAL POSTGRADUATE SCHOOL
Monterey, California

DISSERTATION

A HIERARCHICAL APPROACH TO MULTICAST IN A DATAGRAM INTERNETWORK

by

Robert J. Voigt

March 1996

Dissertation Advisor: Shridhar B. Shukla

Approved for public release; distribution is unlimited.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE March 1996	3. REPORT TYPE AND DATES COVERED Ph.D. Dissertation

4. TITLE AND SUBTITLE A HIERARCHICAL APPROACH TO MULTICAST IN A DATAGRAM INTERNETWORK	5. FUNDING NUMBERS This research was partly funded by the NSF RIA Grant 9309316.
6. AUTHOR(S) Voigt, Robert J.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/ MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/ MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Multicasting in datagram internetworks enables multi-party interactions among users distributed over wide areas by eliminating duplicate packets in one-to-many and many-to-many communication. It requires formation of a tree to distribute multicast data to the communicating group of members.

Present multicast techniques need improvement in scope control, resource discovery mechanisms, and tree construction to efficiently support a large number of global groups with dense as well as sparse membership. We deploy a hierarchy of clustered routers with the following features to make these improvements. Each group is assigned a *scope level* enabling access to resources at that level when members join and permits well-defined boundaries for scope control. The list of border routers and presence of groups at any level is maintained and supplied to members by a level-specific resource discovery mechanism called a *registrar.* To make tree construction scalable, the border routers determine the shortest inter-cluster paths to source clusters using the available unicast routing information, facilitating aggregation of router state for all senders in a cluster. Unlike the existing approaches, administrative configurationof the hierarchy eliminates the need for locating distribution centers dynamically.

14. SUBJECT TERMS Multicast, hierachical, internetwork, registrar, scope control, clusters, CHARM, cluster-based	15. NUMBER OF PAGES 163
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

13. We characterize the path length performance of the proposed hierarchy by providing an upper bound for the penalty as compared to source-specific trees. Simulation results for randomly generated topologies verify the worst case penalty and show the actual penalty to be significantly less. These results show that the proposed hierarchy can be deployed over the existing unicast routing infrastructure to achieve scalable multicasting with the required scope control while keeping the path length penalty bounded. The architecture described permits further improvements in the path length penalty if the identified enhancements to the underlying unicast routing mechanisms are made.

A HIERARCHICAL APPROACH TO MULTICAST IN A DATAGRAM INTERNETWORK

Robert J. Voigt
Commander, United States Navy
B.S., United States Naval Academy, 1979
M.S., Naval Postgraduate School, 1986

DOCTOR OF PHILOSOPHY IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1996

Author: [illegible]
Robert J. Voigt

Approved By: [illegible] [illegible]

Paul H. Moose
Professor of Electrical and Computer Engineering

Herschel H. Loomis, Jr.
Professor of Electrical and Computer Engineering

[illegible] Norman F. Schneidewind

Gilbert M. Lundy
Professor of Computer Science

Norman F. Schneidewind
Professor of Information Sciences

[illegible]
Shridhar B. Shukla
Professor of Electrical and Computer Engineering
Dissertation Supervisor

Approved by: [illegible]
Herschel H. Loomis, Jr., Chairman
Department of Electrical and Computer Engineering

Approved by: Maurice D. Weir
Maurice D. Weir, Associate Provost for Instruction

ABSTRACT

Multicasting in datagram internetworks enables multi-party interactions among users distributed over wide areas by eliminating duplicate packets in one-to-many and many-to-many communication. It requires formation of a tree to distribute multicast data to the communicating group of members.

Present multicast techniques need improvement in scope control, resource discovery mechanisms, and tree construction to efficiently support a large number of global groups with dense as well as sparse membership. We deploy a hierarchy of clustered routers with the following features to make these improvements. Each group is assigned a *scope level* enabling access to resources at that level when members join and permits well-defined boundaries for scope control. The list of border routers and presence of groups at any level is maintained and supplied to members by a level-specific resource discovery mechanism called a *registrar*. To make tree construction scalable, the border routers determine the shortest inter-cluster paths to source clusters using the available unicast routing information, facilitating aggregation of router state for all senders in a cluster. Unlike the existing approaches, administrative configuration of the hierarchy eliminates the need for locating distribution centers dynamically.

We characterize the path length performance of the proposed hierarchy by providing an upper bound for the penalty as compared to source-specific trees. Simulation results for randomly generated topologies verify the worst case penalty and show the actual penalty to be significantly less. These results show that the proposed hierarchy can be deployed over the existing unicast routing infrastructure to achieve scalable multicasting with the required scope control while keeping the path length penalty bounded. The architecture described permits further improvements in the path length penalty if the identified enhancements to the underlying unicast routing mechanisms are made.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ACRONYMS

C_k	A Cluster at level k
$C_k(s)$	A Cluster at level k that contains some node s
d	The shortest path distance
$diam$	The diameter of a network
E	The set of edges for a graph
G	A graph
h	Hop count
h_c	Hop count using clustering
k	The index indicating level of a cluster
m	The number of levels in a hierarchy
N	A network
n	Index indicating level of a multicast, MBR, or group
(SC,G)	A routing table entry for source cluster SC and group G
(S,G)	A routing table entry for source S and group G
$(*,G)$	A routing table entry for group G using a shared tree
V	A set of vertices
ABR	All Border Router group
BR	Border Router
CBT	Core Based Tree protocol
CER	Closest Entry Routing
CST	Center-specific tree
DBR	Designated Border Router
DIS	Distributed Interactive Simulation
DVMRP	Distance Vector Multicast Routing Protocol
EC	Electronic Classroom
FOM	Figure of Merit
HPIM	Hierarchical PIM
GOMR	Group Oriented Multicast Routing Protocol
HDVMRP	Hierarchical DVMRP
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IP	Internet Protocol
LAN	Local Area Network
Mbone	Multicast Backbone
MBR	Multicast Border Router
MOG	Measure of Goodness
MOSPF	Multicast Extensions to OSPF
MST	Minimum Spanning Tree
NMR	Non-membership Reports
OBR	Overall Best Routing

OSPF	Open Shortest Path First
PIM	Protocol Independent Multicast protocol
PIM-DM	PIM-Dense Mode
PIM-SM	PIM - Sparse Mode
RP	Rendezvous Point
RPB	Reverse Path Broadcast
RPF	Reverse Path Forwarding
RPM	Reverse Path Multicast
SMT	Steiner Minimal Tree
SST	Source-specific Tree
TRPB	Truncated RPB
ttl	time-to-live field
UBR	Unicast Border Router
VC	Video Conference

ACKNOWLEDGEMENTS

I would like to acknowledge the financial support received from the National Science Foundation. This research was partly funded by the NSF RIA Grant 9309316.

The work presented in this dissertation is the culmination of support from many people. I am especially fortunate to have been able to work with one of the most dedicated, hard working and patient men I have known, my advisor, Shridhar Shukla. His ability to keep me focussed and willingness to do whatever was necessary, even from a great distance, was instrumental in my completion of this document. I am grateful for his guidance, insight and friendship.

I have also had the honor to work with another hard working and extremely reliable colleague, Bob Barton. It seemed Bob was always there at the right time. I am extremely grateful for his support in the construction of the simulator, the tools needed to sanity check my data and his ear, which I often bent. I also would like to thank Professor Craig Rasmussen, who was willing to donate much of his valuable time just to listen, speak graph theory with me and help me to understand the magnitude of the problem.

I would like to thank my committee members, Professors Bert Lundy, Norm Schneidewind, Paul Moose and Hersch Loomis for their patience, thoroughness and perseverance.

Finally, I would like to acknowledge the omnipresent support, without which, I could never have attained my goal. That support came from my loving wife Patti and my family, Christine, James and Grace. Words cannot express the gratitude I have for their patience, support and love.

I. INTRODUCTION

A. MULTICASTING IN AN INTERNETWORK

1. Background

The integrated packet-switched wide area networks of the future are expected to provide users with a variety of multi-party interaction capabilities. These services will benefit from multicasting for datagrams internetworks, pioneered in [15], permitting elimination of traffic caused by unnecessary packet copies in many-to-many communication. Multicast over internetworks typically requires the network nodes to form a routing tree based on the members of the group and their location.

The capabilities that are enabled by such a multicast service will be useful in the following application areas:

- Distributed databases
- Interactive multi-party interactions such as video conferencing
- Wide area distributed computations and repositories
- Distance learning
- Distributed interactive simulation

For the multicasting over internetworks to be useful across these diverse application areas, it must support the following model of group interaction: dynamic group membership, ability of group members to reside anywhere on the network and best-effort datagram delivery as implemented in the present Internet.

The Multicast Backbone (MBone) is a global multicast internetwork in operation today and is growing rapidly [48]. The MBone is overlaid on top of the present Internet and supports the above model of group interaction. It is an outgrowth of the first two Internet Engineering Task Force (IETF) “audiocast” experiments in which live audio and video were multicast from the IETF meeting site to destinations around the world [9].

2. Mechanisms

A fundamental mechanism required for multicasting over datagram internetworks is the construction of a data distribution tree among group members. There are two basic approaches to this multicast tree construction. The first is a shared or center-specific tree (CST) and the other is a source-rooted or source-specific tree (SST). The center-specific approach utilizes a single tree, rooted at a suitable center, that is shared by all senders. In the source-specific approach, each sender builds a separate tree rooted at itself. A properly constructed CST consumes fewer network resources, such as router memory space and link bandwidth, for an interaction that contains multiple senders. The use of a center provides new members, that are not aware of who the current members are, a destination in the network to communicate with at the time of joining the interaction. Alternatively, the SST approach minimizes the path length metric such as the delay seen by receivers from each sender. When SSTs are used, a way for the new receivers to discover the current senders must be provided. Thus, the choice between CSTs and SSTs presents a trade-off between delay and tree cost, as illustrated in Figure 1.1. Both approaches must provide a way for new members a mechanism to discover resources, *viz.*, the center in case of CSTs and senders in case of SSTs. [13][53]

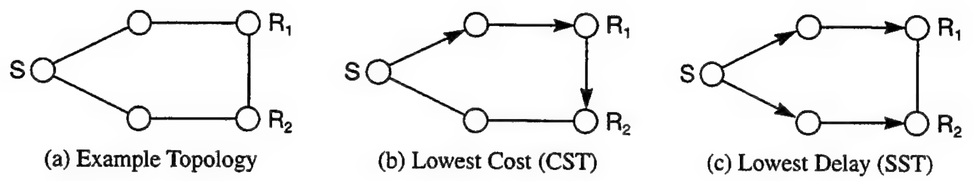


Figure 1.1: SST vs. CST Trade-off

The current IP-multicasting techniques use two standardized protocols called Distance Vector Multicast Routing Protocol (DVMRP) [52] and Multicast Extensions to Open Shortest Path First (M-OSPF) protocol [40] which construct SSTs. DVMRP is the predominant multicast protocol used in the MBone. In addition to the above protocols, two new protocols called the Core-based Tree protocol (CBT) [3] and Protocol-independent

Multicast (PIM) [18] are under consideration by the IETF. The CBT protocol constructs a CST and the PIM protocol provides the end user with the option to join a CST or an SST with respect to a particular sender.

B. CHALLENGES IN MULTICAST

1. Tree Construction and Maintenance

Approaches based on CSTs require the construction of a spanning tree for the members of a group in the network. The construction of an optimal spanning tree with respect to tree cost for a subset of a network is the Steiner Minimum Tree (SMT) problem (see Chapter II, section B.1). The construction of an SMT is a well-known NP complete problem [27]. As such, construction of minimum cost trees for dynamic groups in internetworks using heuristic solutions for the SMT problem is considered impractical. However, centers can be used to facilitate tree construction as well as resource discovery. Both PIM and CBT use centers, called rendezvous points (RP) and cores, respectively, as an aide to construct a spanning tree. The RP and core act as the root of a shared distribution tree. The RP also provides a new receiver with the current list of senders to facilitate the option of joining an SST.

In their present forms, these protocols suggest that the centers may be selected administratively. The center clearly plays a vital role in the quality of the resulting trees. If the center is located far from the participants, a longer packet delay is experienced and excessive resources are consumed. If a good center is selected for a set of participants which later changes drastically, the center may severely impact the quality of the tree for the new participants. Finally, the probability that an administratively located center will be a good one for all participants decreases with the number of participants. [47][51]

After the trees have been constructed, two methods are used commonly for maintaining the tree state in routers, *viz*., "hard state" and "soft state." These refer to whether or not router table entries for a tree are allowed to time out (soft) or if they must be explicitly removed (hard). The advantage of soft state is that the tree is constantly refreshed to take advantage of other lower delay paths as they become available. The disadvantage is that the tree must be refreshed or re-constructed periodically leading to additional processing overhead in the routers and a potential delay in the delivery of the first data packet following a refresh. The advantage of a hard state is that the tree is constructed and removed explicitly. Thus, it is not "data-driven" and does not cause processing delays once the tree is established. The disadvantage is that it is not possible to adapt the tree to the changing delay conditions in the network. [1]

2. Scalability

The two multicasting protocols deployed in the MBone which construct SSTs are Distance Vector Multicast Routing Protocol (DVMRP) and Multicast extensions to Open Shortest Path First (M-OSPF) routing protocol. Both are regarded as suitable for multicasting within a routing domain only as both protocols exhibit occasional network-wide broadcasting behavior. As a result of this broadcast behavior, DVMRP and M-OSPF are not scalable to inter-domain interactions.

The explicit tree construction techniques in CBT and PIM remove this broadcast-behavior problem. However, as both use a center during tree construction, they have the scaling problem of locating a center. Due to the computational complexity of locating a center algorithmically for the inter-domain case, locating these centers administratively has been suggested. However, a single center located in this manner is likely to be a poor choice when group members span the entire topology.

Another significant scalability problem in the inter-domain case is the aggregation of senders and groups. Aggregation refers to the maintenance of combined state corresponding to related collections of senders and groups in the intermediate routers keeping the total state maintained at a router manageable for a large number of senders and groups. A single shared tree which requires on-tree routers to maintain one (*, *G*) entry regardless of the number of senders to the groups provides the best case for sender aggregation as all the senders are aggregated onto one distribution tree. However, a single shared tree for a group spanning the Internet is not a scalable solution for either traffic concentration or path length performance metrics. The aggregation of groups is a more difficult problem which has not been addressed in any of the multicast proposals or implementations. This would encompass the concept of multiple groups sharing one link with a single routing entry for this "group of groups."

3. Scope Control

Presently the "class D" Internet Protocol (IP) address is reserved for multicast (i.e., the most significant 4 bits of the address are 1110) [14]. Since the class D address is not bound to any specific part of the network in the Internet, it is considered to be a "flat" address. The difficulty with this approach is that any multicast, due to the network-wide validity of its address, becomes a potentially global multicast. It is considered undesirable to introduce structure into the multicast address by the Internet community as it requires a change in the installed base of MBone applications and violates the ability of a sender to send to any group from anywhere in the network by simply knowing its address.

Another difficulty presented by a flat address space is that there is no method to scope or restrict a multicast within a boundary based on the address alone. The present solution is to limit the "time-to-live" (ttl) field in multicast packets. The ttl field in the Internet Protocol (IP) has a time unit of seconds but specifies that every router must decrement the field by at least 1 [41]. Thus if this occurs, it actually specifies the number of hops a packet is allowed to travel before being dropped. This makes it difficult to determine the value to be inserted in the ttl field. As the Internet becomes more connected, a small ttl value may reach many more nodes than the originator intended.

C. CONTRIBUTIONS

In this dissertation, we have addressed the above three challenges facing network level multicast by developing a hierarchy of clusters of routers as the framework within which multicast distribution trees are to be constructed. We have developed a Cluster-based

Hierarchical **Ar**chitecture for **M**ulticast, referred to as CHARM throughout this document and evaluated its performance with respect to delay.

The general approach taken in this work is to make run-time tree construction, scope control, and resource discovery scalable by using an administratively configured hierarchy of routers. The overall contribution made by this work is to propose a hierarchical multicast architecture and show that the proposed hierarchy can be deployed over the existing unicast routing infrastructure to achieve scalable multicasting while maintaining a known worst case path length penalty and actual path lengths that are very close to the shortest paths.

The specific contributions made by this work are listed below.

- The use of a *scope level* for each group enables access to resources at that level when members join and permits well-defined boundaries for scope control. It also permits the group address to remain flat.
- The list of border routers and presence of groups at any level is maintained and supplied to members by a level-specific resource discovery mechanism called a *registrar*. The organization and functions of this entity are described in detail. The need for such an entity has been stated[8], however, none of the existing multicast protocols has addressed it.
- Tree construction is made scalable by making the border routers determine the shortest inter-cluster paths to source clusters, based on the available unicast routing information, facilitating aggregation of router state for all senders in a cluster. Existing protocols require construction of shortest paths to individual senders creating scaling problems.
- The need for locating distribution centers dynamically, a major drawback of existing protocols, is eliminated by an administrative configuration of the hierarchy. This also permits incorporation of suitable management policies.
- The path length performance of the proposed hierarchy has been characterized by providing an upper bound for the penalty as compared to source-specific trees. Simulation results provided for randomly generated topologies verify the worst case penalty and show that the actual path lengths can be made to approach those given by SSTs by sacrificing the granularity of scope control.
- Finally, methods have been suggested to achieve further improvements in the path length penalty by identifying the enhancements required to the underlying unicast routing mechanisms.

D. DISSERTATION OVERVIEW

This dissertation is organized as follows.

In Chapter II, we discuss current and related work in the area of multicasting for datagram internetworks and summarize the properties of different protocols with respect to the features desirable in a multicasting techniques suitable for global deployment.

In Chapter III, we introduce CHARM and describe its components in detail including the actions taken upon joining of senders and receivers from the different parts of the hierarchy. We illustrate the concepts behind the organization using a running example of a small three level topology. We also identify all the assumptions we make in terms of the unicast routing required and the properties of the clustering assumed.

In Chapter IV, we analyze the proposed architecture with respect to the path lengths obtained. We arrive at the most suitable approaches to route data inside the source cluster, inside the receiving cluster, and between the clusters by describing the alternatives and their implications in detail. Once the most suitable techniques are identified, we describe how the worst case path length penalty between any sender and receiver is guaranteed.

In Chapter V, we describe how additional infrastructure in terms of hierarchical unicast routing and collection of a measure of goodness permit improvement in the worst case path length penalties in the receiving and sending clusters respectively.

In Chapter VI, we describe the network overhead resulting from the deployment of CHARM in terms of the control traffic and the amount of state maintained at the routers.

In Chapter VII, we provide path length results for three different types of multi-party interactions on two types of topologies generated using random graphs. An analysis of the results is provided showing exactly how the penalties are distributed among the senders and receivers as the size of clusters in a hierarchy varies.

In Chapter VIII, we provide concluding remarks and suggestions for future research.

II. RELATED WORK

In this chapter, we examine the fundamental work on which the existing multicast protocols and proposals are based. We then survey the protocols and proposals and evaluate them with respect to the requirements of a globally scalable multicast.

A. TECHNIQUES

Multicast protocols are based on the work done on multi-destination delivery by Dalal and Metcalfe [12] and center-specific trees by Wall [53]. The results and techniques reported in these have led to the development of protocols based on source-specific and center-specific trees. Other areas of research examine the distributed Steiner tree problem [22, 35] and solutions for using multicast in specific application areas[33].

1. Reverse Path Forwarding

Dalal and Metcalf introduce the concept of Reverse Path Forwarding (RPF) as a routing method to reduce the number of duplicate copies of broadcast packets in a network. Their technique forwards a broadcast packet on all links except the one it arrived on, provided the one it arrived on is along the shortest path back to the sender. If it is not, the packet is dropped. Use of this method does not eliminate duplicate packets, rather it limits them and prevents the flooding from continuing indefinitely.

Figure 2.1 shows how RPF works in a series of illustrations. Each graph shows an iteration of RPF for a broadcast by the node marked **S**. In this example, every node, except nodes **c** and **d,** receives two copies of the data, nodes **c** and **d** receive three. After only three iterations the broadcast has been delivered to all of the members and after four, it is complete. Using RPF, no additional information about the network other than shortest path distances between two nodes is required. No additional information is stored at a node and therefore no additional data structures are required beyond the standard unicast routing tables.

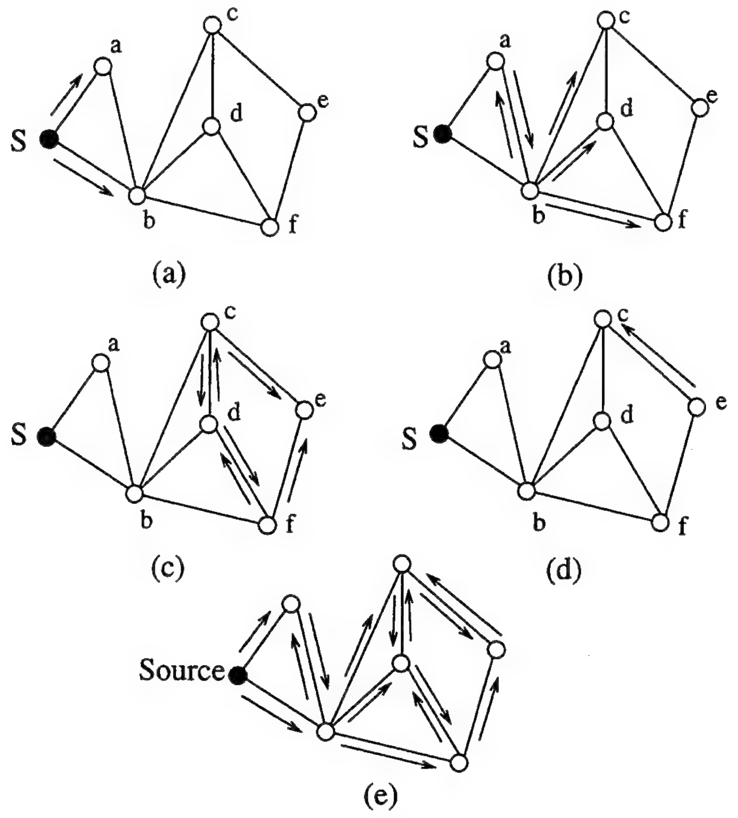


Figure 2.1: An Example of Reverse Path Forwarding

The algorithm is considered to be practical for broadcast and optimal, in that data is delivered on the shortest path, if delays are symmetric on all links [12]. RPF is considered to be an efficient method for broadcast, where all nodes are involved. It is unsuitable for a subset of the nodes since nodes that are not intended recipients still receive packets.

2. Reverse Path Multicast

Multicast capability has existed at the local area network (LAN) level using the LAN shared medium [28, 29]. Deering and Cheriton introduced multicasting as an efficient multi-destination delivery in an internetwork in Reference [15]. They also introduced the concept of an unknown destination delivery using a group address. The group address, similar to a broadcast address in that it has more than one intended recipient, restricts the recipients to some subset of the network. The reverse path multicast (RPM) work adds a number of refinements to the work done in Reference [12] making it more suitable for an internetwork. We examine the work done in References [13] and [15] as it is the basis for the present internet multicast protocols. RPM is presented as series of enhancements to RPF below.

The first enhancement to RPF is that the group address identifies a host group to receive the packets sent to that address[10]. The sender need not know the membership of the group. Two types of groups are possible - *open* and *closed.* In a closed group, a sender is required to be a member in order to send to the group. In an open group, the sender is not required to be a member [15]. As in References [13] and [15], we focus on the less restrictive open group in this work.

Deering and Cheriton specify extensions to two existing distributed unicast routing algorithms, namely, *distance vector* and *link state*. Distance vector routing requires that each node maintain the distance from itself to each possible destination. It does this by gathering information from its neighbor's distance vectors[41]. In link state routing, each router keeps a complete map of the topology and computes routes to each destination. It does this by sending advertisements to all routers updating the network topology as it changes. Distance vector routing is described in References [26] and [37]. Link state or Open Shortest Path First (OSPF) routing is described in Reference [39]. The set of extensions to these unicast routing protocols have led to the Distance Vector Multicast Routing Protocol (DVMRP) and Multicast Extensions to OSPF (M-OSPF). One of the key features of these protocols is that they rely on the unicast routing tables stored in the routers to construct their multicast routing tables.

The second enhancement to RPF is that multicast at the internetwork level stores multicast specific routing data in routing tables. At each router, these tables store a subset of the links upon which multicast packets for a group are to be forwarded. Similar to source-based forwarding discussed in Reference [12], shortest path trees are formed from each sender to all members of a group. This is accomplished using a data driven method which is described below. The method constructs the trees and fills in the routing tables based on where the data is *not* supposed to go. In essence, the data is flooded or broadcast out using RPF and then links are "pruned" back as they are not needed.

The construction of the multicast tree is best understood in terms of the incremental improvements made to RPF. Reverse path broadcast (RPB) refers to RPF modified by identifying "child" links and eliminating duplicate data on LANs with more than one attached router [15]. Only one of the attached routers is designated to forward packets onto the LAN. A further modification called truncated reverse path broadcast (TRPB) stops the multicast packets from being forwarded onto a LAN in the case where no members of a group exist by sending a prune message back towards the source of the multicast [15]. This effectively prunes the "leaf" from this branch of the tree. In a final refinement, reverse path multicast (RPM), the prune message is in the form of a non-membership report (NMR) [15]. If intermediate routers receive NMRs from each of their outgoing links, they in turn generate an NMR and send it back up the tree towards the source. The NMR reports prune the tree back to only those routers who have members or are on the path to other routers who have members attached.

The above strategy, used in DVMRP, exploits flooding and subsequent pruning to shape the multicast tree. This method constructs a lowest delay tree; however, the flood and prune strategy, with its occasional broadcast behavior, is not considered a scalable internetwork-wide solution.

3. Center-specific Trees

The center-specific tree (CST), an extension of the work by Dalal [11], proposes to provide a more efficient method of broadcast and selective broadcast (i.e. multicast) [53]. The work is based on the use of a minimum spanning tree (MST) as a delivery mechanism. An MST is a single minimum-cost tree which spans all nodes. The cost of computing an MST is a well-known graph theoretic problem and many solutions exist [6]. The problem of computing a minimum cost tree for a subset of the graph, is known as a Steiner minimum tree problem (see section B.1).

In [53], Wall proposes a new technique for constructing a center-specific tree based on locating a center of the network called center-based forwarding. This method for constructing a single shared tree attempts to strike a balance between network cost and delay as discussed in Chapter 1, section B.1. Minimal cost and minimal delay cannot be achieved using one type of distribution tree [53].

Wall's result is significant in that it shows the maximum delay bound of a center-specific tree to be twice that of a shortest path tree. This result is used in Reference [1] to build a multicast protocol known as Core Based Trees. The construction of the center-specific tree does not use flood and prune but rather an explicit join mechanism with no flooding properties. An explicit join is characterized by a unicast message which results in a branch being added to a tree. The low tree construction and storage costs of this approach is potentially useful for a global internetwork-wide multicast for scaling purposes. We use the center-specific tree in the hierarchy to multicast non-delay-critical control messages.

B. ISSUES IN TREE CONSTRUCTION

Each of the above multicast techniques involves the construction of some type of delivery tree. We examine the issues related to constructing and maintaining trees for multicast.

1. The Steiner Tree Problem

As discussed in the previous section, a single shared tree among nodes provides for a low cost solution to either a broadcast or a multicast distribution problem. The difference between these two problems is that a minimum cost tree that spans all of the nodes (broadcast) is a minimum spanning tree, while the minimum cost tree spanning only a subset (multicast) is a Steiner minimum tree (SMT) [27].

The difficulty with constructing an SMT is that it is an NP-complete problem [31]. Several heuristic approaches have been suggested to find a good solution in References

[22], [27] and [35]. These approaches are not suitable for the multicast model of the internet, that of open groups and dynamic group membership [1]. Even the center-specific tree construction method developed in Reference [53] needs an initial center which is used as a focal point for constructing the tree. The location of a center with respect to the group members, affects the quality of the resulting tree [47]. While it is possible to get a good initial center, dynamic membership changes drive a need for greater flexibility that cannot be provided by these solutions.

2. Tree Reconfiguration

The heuristic approaches applied to construct a minimum cost shared tree assume some initial member distribution. It follows then that the quality of the tree will likely change as members join and leave the group. The work done in Reference [19] attempts to answer how the tree quality gets affected with membership changes.

There are three potential solutions to handle dynamic membership. One is to re-compute the tree after each change, another is to make modest alterations to an existing spanning tree [54]. For example, in Reference [55], an algorithm was applied to modify the delivery tree dynamically by adding branches only when new members join and removing them only when members leave. Re-computing the tree after each change is an expensive solution leading to disruptions to members who remain in the group. A third solution is to build a sub-optimal tree which is resilient to change [19].

It has been shown that suboptimal trees do not perform too poorly [19, 54, 55]. They show that while there are spikes of poor performance for certain members of the group, overall, the average performance does not get substantially worse using either a slow re-configuration or a sub-optimal tree.

These results are potentially useful for a hierarchical multicast that may require more than one tree to accomplish a multicast. They show that a periodic refresh is sufficient to maintain tree quality and frequent dynamic reconfiguration only increases the amount of processing and storage overhead while not improving the path lengths significantly.

3. Center Location for Shared Trees

The problem of locating a center for the center-specific tree construction protocols is examined in References [46], [47] and [51]. Prior work has focussed on evaluating performance of randomly placed centers. Center location is also discussed in References [1] and [53] but no working protocol is offered. In References [46], [47], and [51], it is shown that with some intelligent placement, the performance of the center-specific tree can be improved. These approaches again offer heuristic solutions to solve this problem since to find the optimal center is an NP complete problem.

In the hierarchical multicast architecture we propose, we note the value of the center-specific tree from a tree-cost viewpoint. We believe that it is not practical, for a global multicast solution, to incur the potential path length penalty possible when a center-specific tree is used for the data distribution portion of the hierarchy. This penalty is particularly significant as the imposition of a hierarchy will incur penalties of its own. Furthermore, we show that the deployment of the hierarchy reduces the need for another mechanism for sender aggregation that is provided by a shared tree.

C. EXISTING IMPLEMENTATIONS

Existing multicast protocol implementations can be categorized as intra- and inter-domain multicast protocols. The primary distinction between the categories is the scalability improvements in the inter-domain protocols. Another important distinction is to make the inter-domain protocols independent of the underlying unicast protocols. The protocols to accomplish the two types of multicast are discussed below.

1. Intra-domain Techniques

The intra-domain protocols have been implemented and in use for several years.

a. DVMRP

The present IP multicast deployed in MBone is DVMRP [52] which is based on work done by Steve Deering [14] where the routers use the distance vector method for unicast route calculation. DVMRP constructs a source-specific tree for each sender in a group, by maintaining an entry for a source, group (S,G) pair.

In DVMRP, the first multicast data packet is flooded out using RPM as discussed in section B.2 above. The tree is formed through pruning based on the NMRs of non-participating routers. This flood and prune strategy is the primary reason why DVMRP is not considered a scalable method for multicast tree construction. Flooding of data is not a desirable feature in an internetwork environment, because it creates unnecessary overhead for un-involved nodes.

b. M-OSPF

The approach based on OSPF unicast routing is called Multicast Extensions to OSPF (M-OSPF) and is described in Reference [40]. OSPF is a link-state routing protocol which provides routers a link-state database describing the network topology and updates the database through the use of advertisements. In M-OSPF, a new OSPF advertisement is added describing multicast locations [40]. Like OSPF link state changes, group information is broadcast across the network so that each router can maintain their link-state database.

M-OSPF can only be run in a network running OSPF. All M-OSPF routers have a complete topology map of the location of group members. The broadcast nature of the link state advertisement and the storage required for a global membership information are the main reasons why this approach is not considered a scalable internetwork-wide solution.

2. Inter-domain Techniques

The primary changes for the inter-domain techniques is the addition of shared trees, unicast protocol independence, and the addition of explicit joins and leaves to reduce or eliminate the flooding behavior of the intra-domain protocols in tree construction.

a. PIM

Protocol Independent Multicast (PIM)[16] is one of the two draft standards being considered by the Internet Engineering task Force (IETF) to solve the scalability problem. The other is CBT, described below. PIM is designed to be independent of the underlying unicast protocol. The PIM architecture takes advantage of the existing hierarchy in the network and introduces two modes, dense and sparse. These two modes address the type of interaction desired relative to group composition and the size of the network. A dense group has a group membership that is "densely distributed across an internet" [17] and, conversely, a sparse mode group has a membership which "may span wide-area (and inter-domain) internets"[16]. Dense mode PIM (PIM-DM)[17] is similar to DVMRP and is a source-specific tree based protocol. This method was found to be undesirable for groups whose members are distributed sparsely across a wide area. Sparse mode PIM (PIM-SM) [16] uses a center-specific tree construction designed to address the scalability of its dense mode counterpart. The distribution center of PIM-SM is called a Rendezvous Point (RP). PIM-SM multicast tree construction also allows a hybrid mode, which includes both center-specific and source-specific trees, when the receivers request it [18].

PIM tree construction revolves around the selection of the rendezvous point (RP). All senders for a group must register with the RP in the network. Receivers requesting to join the group set up the path from themselves to the RP. This is also how they learn about the senders in case they later desire to form a shortest path tree between them and eliminate use of the RP for that source and receiver pair.

b. CBT

Core Based Trees (CBT) is a protocol for multicast tree construction which also uses a center-specific or shared delivery tree [1]. CBT is similar to PIM-SM in that they both initially choose a center from which they build the tree, however, in CBT, multiple centers are allowed for one group. One of the primary differences between PIM(SM) and CBT is that CBT maintains "hard state" while PIM uses a "soft state." A soft state approach is data-driven, trees are built as needed and state for the trees is allowed to time out. The hard state approach does not time out tree information after a period of inactivity.

A consequence of the hard state is that multicast tree branches do not adapt to unicast route changes [1]. This is good for uninterrupted packet flow since the routes stay constant, but it may lead to sub-optimal branches on the tree. Also, soft state trees tend to have a higher join latency since the tree state may time out where a new branch might graft. We show that both of these approaches can be applied in our hierarchical multicast.

3. Hierarchy-based Proposals

a. Hierarchical DVMRP

Hierarchical DVMRP (HDVMRP) is a proposed solution to address the exponential growth of MBone. It consists of a two level hierarchy which runs separate instances of DVMRP at the two different levels. The level 1 regions are composed of one or more Internet domains. The regions contain one or more boundary routers which are responsible for connecting regions. All boundary routers run two levels of multicast protocols where the level 2 protocol is an inter-region protocol. The result is that the protocol uses a flood-and-prune strategy at the inter-domain level. This yields the benefit of shortest path trees at the inter and intra-domain levels. [49]

The primary problem is that the flood-and-prune nature of DVMRP continues to be the basic tree construction strategy of this protocol. The number of nodes involved in flood-and-prune is reduced in that the level 2 trees are only between border routers. However, in order to construct the tree, the first level 2 packets are flooded throughout a region. In addition, the encapsulation of packets at level 2 leads to duplication of packets inside a level 1 region.

We view this proposal as a specific incremental improvement to an existing protocol. It solves some of the scaling problems of the MBone by only flooding regionally and less often than pure DVMRP. While the concept of regions is similar to the clusters proposed in this hierarchy, the use of the concept is quite different. We do not use any flooding-based tree construction and we do not allow the duplication of packets inside a cluster to arrive at the same destination.

b. Hierarchical PIM

Hierarchical PIM (HPIM) is a proposal to solve very specific problems with PIM, that of the advertisement of rendezvous points (RPs) to group members and mapping RPs to groups [24]. The proposal consists of a control hierarchy for RPs in which the RPs themselves are structured in a hierarchy, not the members.

In HPIM, the members join towards the lowest level RP. If the join is meant to go to a higher level scope, then that RP forwards the join request to the next level RP. The RPs are structured in a hierarchical fashion to allow for increasing scope of the multicast. In PIM, the senders unicast to the RP which then multicasts out on the shared tree to all receivers. In HPIM, the data is forced to flow along the RP tree from RP to RP at each level. At each RP it is de-encapsulated, checked for receivers and then re-encapsulated and forwarded on to the next level RP. If receivers exist at a level, the RP forwards the data out onto the tree. The problem this solves for PIM is the case of the nearest RP which is not local but the group is local. The flat version of PIM forced multicast traffic outside of the users desired scope to get to the nearest RP. The advantage of PIM (and HPIM) is that members who have bad paths via their RP for senders who have a high data rate have the option of switching to a shortest path tree.

We view this proposal as a specific incremental improvement to an existing protocol. It solves the problem of PIM RP discovery, but it introduces a new level of complexity and a new set of problems, primarily that of the data forced to flow along what may be a very sub-optimal path from RP to RP.

c. Hierarchical Multicast

A hierarchical multicast routing algorithm is proposed in Reference [57] which included clustering the network into a two level hierarchy. The clusters are the first level of the hierarchy. One of the nodes of each cluster is selected as the local core. The cores form the second level of the hierarchy.

The cores form a shortest path tree rooted at themselves to all of the members in their cluster and another to all other cores. Senders send traffic to the core which multicasts it out on both levels, one for local traffic and one to the other cores on the tree. The authors call this a group oriented multicast routing (GOMR) protocol.

GOMR performs better for smaller cluster sizes but has a problem with scalability since it only supports 2 levels. The larger cluster sizes scale better but there is a trade-off in end-to-end delay. Three separate protocols are introduced in Reference [57], of which one is for hierarchical multicast tree construction. The others are for set-up and maintenance of the hierarchy.

The first protocol determines neighbor sets to dynamically cluster the network. It does this through the broadcast of a probe message to determine how far away members are allowed to be from the core. This, in opposition to the resulting multicast, does not scale for large cluster sizes. In addition, we have seen that broadcast behavior, even a limited broadcast behavior, is undesirable in an internetwork. Finally, the amount of overhead in time to compute the clusters is linear with the size of the group. This is an unacceptable start-up cost

The third protocol is used to adjust the cores once the group is established to insure fairness among the members of the group. This requires all members of a group to know all clusters and the local cores [57]. Again this is a scaling problem for a large number of small clusters, which are required for better performance.

The tree construction protocol simulation shows good performance characteristics for large numbers of small clusters, however, this solution does not scale. Large clusters have both a scaling and performance problem. The fact that it only supports 2 levels makes it unsuitable for scope control with a finer granularity. Our proposal permits a multi-level hierarchy with similar performance traits and greater flexibility in clustering because of the multiple levels. The main advantage is that we do not introduce the broadcast behavior, high start-up costs, and performance problems of a single entry and exit point for a cluster which is required by GOMR. Also, we do not have the inherent scalability problems that this proposal has.

D. EVALUATION OF EXISTING APPROACHES

Table 2.1 lists the criteria for evaluating the above proposals and implementations for scalability, scope control, center location, use of a flat address space, and aggregation.

Table 2.1: Desirable Attributes of a Multicast Protocol

Attribute	Interpretation	Significance
Scalability (**S**)	Handles many dense as well as sparse wide-area groups	Requirements for any internet level multicast protocol
Scope Control (**SC**)	Well-defined boundaries for data and membership forwarding	No overhead for uninvolved nodes
Path Length (**PL**)	Comparison of paths generated with the shortest path source-specific trees	Primary metric for performance characterization
Center Location (**CL**)	Protocol requires a center for resource discovery or tree construction	Impacts the performance of the resulting tree
Flat Address Space (**F**)	Every multicast can be global in scope	Addresses cannot be used for scope control by themselves
Aggregation (**A**)	Routers can reduce the state maintained by combining many individual entries	Permits handling of senders and/or groups without excessive memory/update requirements

In Table 2.2, we show how each of the multicast schemes can be characterized by the above attributes:

Table 2.2: Summary of Existing Multicast Protocols

Protocol	Principle Feature	S	SC	PL	CL	F	A
DVMRP	Shortest paths via multiple trees	✗	✗	✓	--	✓	✗
M-OSPF	Requires OSPF	✗	✗	✓	--	✓	✗
PIM	Shared tree/SST Hybrid	✓	✗	✓	✗	✓	✗
CBT	Shared trees with multiple cores	✓	✗	✗	✗	✓	✓
HDVMRP	Hierarchical version of DVMRP	✗	✓	✗	--	✓	✗
HPIM	Hierarchical RP selection.	✓	✓	✓	✗	✓	
GOMR	2 Level Cores	✗		✗	✗	✓	

✗ = not supported, ✓ = supported/support possible, “ ” = unknown, -- = not required

Some of the above table entries are proposals, such as HPIM, HDVMRP and GOMR. As a result, the exact status of any particular point may not be known at this time.

The initial multicast solutions do not scale due to broadcast behavior in the protocols. The proposals to address scalability have introduced new problems of their own. For example, since they rely on center-specific trees to solve the broadcast problem, they have introduced a center location problem. None of the proposals has provided complete solution to the scope control problem beyond an administratively scoped address which imposes a simple two level hierarchy through address filtering. None of the solutions has proposed aggregation of groups to simplify handling of a large number of groups and only CBT permits aggregation of senders. Several of the above proposals are beginning to examine the use of a hierarchy, but only as an incremental improvement to problems in existing solutions.

E. Focus of Present Work

The proposed cluster-based hierarchical architecture solves many of the problems discussed in the previous section. It does not propose an entirely new protocol suite from the ground up. Rather it exploits the existing protocols in a structured hierarchical architecture.

We have developed this cluster-based hierarchical architecture for network level multicast (CHARM) and evaluated its performance with respect to path length. We note our basic objectives and focus areas.

- *Scope Control* - To include a comprehensive solution for controlling the scope of data flow as well as flow of membership information.
- *Performance* - The worst case path length penalty must be known depending upon the scope of the group.
- *Scalability* - It must be possible to perform sender aggregation and facilitate group aggregation.
- *Center Location* - There must not be a need to locate centers dynamically and in a group specific manner.
- *Flat Address Space* - The group address must remain flat and the scheme must permit use of name and address resolution schemes.

III. CHARM: A CLUSTER-BASED HIERARCHICAL ARCHITECTURE FOR MULTICAST

We have noted in the previous Chapter that a hierarchy can be used for multicasting in internetworks to address scope control and scaling issues. For illustration, we draw an analogy with the hierarchy in the postal delivery system. Using the initial digits of the ZIP code, a letter can be quickly identified to have a local delivery destination. Thus, the post office can process mail locally instead of regionally and the scope of the letter can be quickly identified. The scaling issue is addressed by the ability to have many local offices which relieve the load on the regional offices. These in turn, relieve the load on the national distribution points and so on. In addition, the intermediate routing points need not know the exact location of the final destination, only its approximate location. The finer detail of the address, such as the street number, is not required until the letter gets near its destination. This simple example shows how a hierarchy is deployed to address the issues of scalability and scope control. We note, however, that the ZIP code represents a hierarchical address and in multicasting over internetworks, we need to maintain the non-hierarchical nature of the group address. The ZIP code system bears a strong analogy with hierarchical unicast routing. We begin this Chapter with a description of such a scheme and relate it to the proposed scheme for multicasting.

A. HIERARCHICAL UNICAST

The hierarchical unicast solution proposed in References [21] and [32] is primarily aimed at reducing the number of routing table entries and the number of updates required to keep the tables current. Given the growth of the Internet at the time, it was determined that storage and updates of routing information would soon become prohibitive in a flat addressing scheme. We explain this solution briefly below.

Consider a network of N nodes where each node has a routing table. The unicast tables have a number of entries that contain the destination address, the delay to that destination, the outgoing interface and the hop count. The outgoing interface, or next node entry, is the link upon which traffic is forwarded to get to the destination along the shortest path. The shortest path normally refers to the minimal delay path, not hop counts. Hop counts are used to determine link or node failures in the network. The problem addressed by the hierarchical solution of [21] and [32] is the following: given a fixed number of nodes in a network, *N*, some number of levels in a hierarchy, *m*, and some number of nodes at each level, *n*, minimize the number of routing table entries for each router in that network.

In Figure 3.1, we show a simple flat topology with *N*=18 nodes to illustrate the reduction in the number of router table entries when a hierarchy is used. We assume that a distributed routing algorithm, such as [26], allows for the exchange of routing information between neighbors. When this information has been exchanged for some period of time, each of the nodes contains an entry in its routing table for each of the other nodes. This entry contains the information about the shortest path to each destination. If we include a self-entry, then each node contains 18 entries in its unicast routing table. The size of this table in a flat topology is O(*N*), which in a large network, would be prohibitive to maintain.

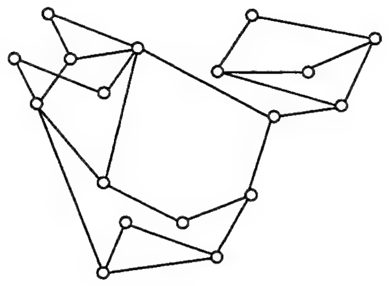


Figure 3.1: A Simple Flat Network Topology

The solution to this problem is very similar to the mail delivery analogy mentioned above where the postal address is a unicast destination. The unicast destination is identified by a single address, however, its address can be aggregated with others who are in the same destination network. The imposition of a hierarchy on a flat topology as shown above implies clustering of nodes. Clusters permit aggregation of individual addresses inside a cluster into one address, thereby reducing the number of entries in the routing tables of the individual nodes. A clustered version of the above topology is shown in Figure 3.2. The clusters are numbered using a decimal notation system. Nodes are defined as 0th level clusters. 0th level clusters are grouped into 1st level clusters which in turn are grouped into 2nd level clusters. The top level cluster contains all of the nodes in the network.

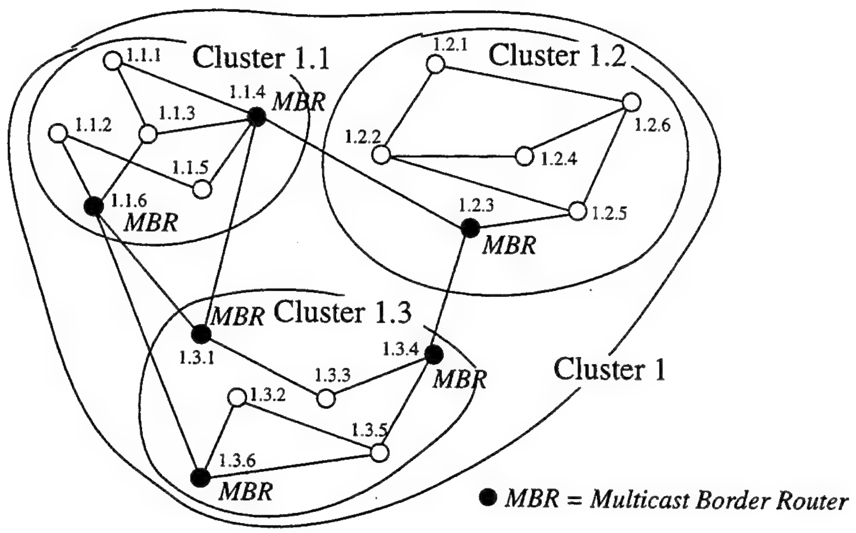


Figure 3.2: Imposition of a Hierarchy on a Flat Network Topology

A sample router table for node 1.1.2 of cluster 1.1 in Figure 3.2 is shown in Table 3.1 below.

Table 3.1: Sample Unicast Routing Table [32]

Destination	Next Node	Delay	Hop Count	
1.1.1				
1.1.2	Self Entry			
1.1.3				**0th Level Cluster Entries**
1.1.4				
1.1.5				
1.1.6				
1.1	Self Entry			
1.2				**1st Level Cluster Entries**
1.3				
1	Self Entry			**2nd Level Cluster Entry**

Note that the routing table now has 10 entries instead of 18 since all addresses for clusters 1.2 and 1.3 are aggregated into one entry each. Given this example, it can be seen how a simple hierarchy saves almost 50% of a unicast table's size by aggregating the addresses.

References [21] and [32] show, that through clustering and address aggregation, significant savings in routing table space can be achieved at the cost of a path length penalty. By routing traffic to a node based on its destination cluster, traffic for all nodes in that cluster follows the same path. This may not yield shortest paths for all members in the destination cluster. In addition to minimizing the router table size, References [21] and [32] also examine this path length penalty. We start by stating their assumptions.

1. Assumptions for Hierarchical Unicast

The path length penalty examined in References [21] and [32] is based on several assumptions resulting from the routing table minimization problem. In what follows, graph G represents the network N under consideration.

Assumption 3.1: Graph G is a connected graph with a set of vertices, V, connected by a set of edges, E. This graph is not directed and the weight on every edge of the graph is unity. Therefore, $G = (V, E)$. G represents the network, N, and vertices represent the network "nodes" or routers.

Assumption 3.2: There exists, at each node in the graph, a routing table which is assumed to contain shortest path routing information for the other nodes in the graph, including the outgoing interface to reach a destination node along that path. This is the unicast routing table.

The assumptions that follow have been made to permit formulation of the routing table size optimization in Reference [21].

Assumption 3.3: The underlying m-level hierarchical clustering structure of the network nodes is such that all clusters at level k, C_k, are of equal degree, n_k, k=1,...,m. The degree of a k^{th} level cluster, C_k is defined as the number of k-1^{st} level clusters included in

C_k. Also, the subset of nodes composing a cluster at any level and their incident edges constitute a 1-connected cluster subnetwork (at *least* one path exists between any pair of nodes).

The two implications of Assumption 3.3 are that clusters are of equal size at the same level and that a node cannot belong to a cluster if it does not have at least one path, internal to that cluster, to another node in the cluster. In the work we present in this thesis, sizes of clusters need not be the same. We do consider the 1-connectedness of clusters essential to our architecture (see Section C.2).

Assumption 3.4: The diameter, which is the maximum shortest path between any pair of nodes, of any k^{th} level cluster is less than or equal to a quantity $diam(C_k)$, $k = 1,...,m$. $diam(C_m)$ represents the diameter of the entire network, $diam(N)$ and

$$diam(C_k) > diam(C_{k-1}) > 0 \quad \forall k \tag{3.1}$$

This assumption states that, based on Assumption 3.3, when all of the clusters at the same level are the same size, it is not possible to have a parent cluster in the hierarchy with a diameter smaller than any of the clusters at the level below. We find, through experimentation on random graphs that are more representative of the internetworks today, that this is not necessarily true and may be difficult to enforce on a real internetwork.

Assumption 3.5: A cluster at any level $k = 1,2,...,m$ contains *a* shortest path between any given pair of nodes which belong to that cluster.

This assumption states that traffic routed between two nodes in the same cluster must be routed internally to that cluster and that the internal routing contains the shortest path between these two nodes.

2. Kamoun and Kleinrock Result

Using the above assumptions, Kleinrock and Kamoun show that large savings can be gained by introducing a hierarchical clustering structure into the unicast routing problem. The length of the routing tables can be reduced from N entries, to $e \ln N$ entries[32]. The path length penalty to be paid for this saving can be as much as two times the shortest path.

Further, that in the case of hierarchical unicast, the path length penalty approaches zero as the size of the network goes to infinity. This is described as:

$$N \to \infty \Rightarrow \begin{cases} h_c/h \to 1 \\ l/N \to 0 \end{cases} \tag{3.2}$$

where h_c is the path length in the hierarchy, h is the shortest path and l is the router table length. Thus, a significant reduction in router table entries with essentially no increase in path length for large networks is achieved for large networks.

B. UNICAST vs. MULTICAST USING A HIERARCHY

The relationship between flat and hierarchical multicast is different from that between flat and hierarchical unicast. Group addresses are global or "flat," and the address effectively aggregates all members of a group into a single address. Thus, the only savings in router table space comes from source address aggregation, unlike the unicast destination address aggregation. Consider the postal delivery analogy. We now have one ZIP code which translates to several destinations that are not geographically close. As a delivery mechanism, the post office does not even know who or where the receivers are unless the receivers subscribe to the ZIP code or the sender supplies a list of receivers. Simply using a ZIP code-like mechanism appears insufficient for multicast. A better analogy for multicast is cable television. The cable company has one cable on which information from many sources is sent to many receivers. It is the receivers' responsibility to select the desired channel. Aggregation is achieved by grouping many sources together on a single distribution channel and letting the receivers sort out which senders they wish to receive. Along the way, some senders' data streams may be diverted to other links or dropped if no longer subscribed to. Thus the need for the receiver's responsibility in the multicast case is an important point of distinction.

In unicast, the hierarchy permits a node to keep information in its tables about those nodes in close proximity while aggregating information about destinations further away. The group address and its intrinsic aggregation of receivers' addresses is "optimally aggregated" for a single group using a center-specific tree. As such, adding members does not increase the routing storage requirements for this group since entries are stored per group. Making multicast addresses hierarchical implies a different and unique address for a group at each level in the hierarchy. Any increase in the number of ways to address a group, which leads to more addresses per group, results in an increase in router storage requirements for the group. It has been stated that, due to a lack of structure in present multicast addresses, group aggregation is not feasible [1]. As any two groups are likely to have membership that is in non-overlapping parts of the network, it is not known how group aggregation could be handled even with hierarchical group addresses. The dynamic nature of groups makes group aggregation even more difficult. Although it appears feasible to aggregate groups if the address were structured hierarchically, by itself it does not appear sufficient for multicasting. Thus, the aggregation deployed for multicast needs to be different.

Scope control, one of the motivations for this work, also works differently for multicast. Since group addresses are to remain global, use of addresses to perform implicit scope control, as in the ZIP code, is not possible. The only implicit method, the use of the ttl field is unrealistic as it cannot be applied in a structured fashion. Thus, explicit scope control appears to be the only practical solution. By assigning a scope level at group creation time, we attempt to save on router table space for those nodes who have nothing to do with the multicast. In addition, we obtain scaling benefits by keeping local traffic local.

C. TERMINOLOGY

We decompose our network into a vertical arrangement of subsystems called clusters. We first define the terms below.

1. Network

<u>*Definition 3.1:*</u> A ***network*** is a connected graph, $G = (V, E)$, with $V = \{v_1, v_2, v_3, \ldots v_n\}$ and $E = \{e_1, e_2, e_3, \ldots e_m\}$ [23].

Definition 3.2: A ***cut-set*** of a connected graph is a set of edges whose removal would disconnect the graph [23]

Definition 3.3: A ***cut*** of a network, N, is a cut-set of the underlying graph, G. A cut partitions V into two subsets, P and $\bar{P}$ such that $P \cap \bar{P} = \varnothing$ and $P \cup \bar{P} = V$ [23].

The physical interpretation of the graph is of an internetwork in which nodes represent routers.

2. Clusters

Definition 3.4: A ***cluster***, C, is a subset of V corresponding to a cut of a network N.

Definition 3.5: All nodes of C with edges to nodes of G not in C are defined as ***border routers*** of C.

Definition 3.6: Nodes of G are defined as ***level 0 clusters***. The set V_0=V of all level 0 clusters and set of E_0 = E of edges between them define graph G_0. Note that $G_0 = G$.

Definition 3.7: A ***level n*** ($n > 0$) ***cluster***, C_n, is a cut in a graph, G_{n-1}, whose nodes are level n-1 clusters and whose edges, e' ($e' \subseteq E$), connect level n-1 clusters.

Definition 3.8: Border routers of C_n are ***level n border routers***.

The physical interpretation of a level 1 cluster is a routing domain. Border routers of level 1 clusters are domain border routers [43, 44]. A level 1 cluster is also referred to as a ***leaf level*** cluster.

3. Logical Hierarchy

The proposed hierarchy is an m level hierarchy, where m is the highest level and is the entire internetwork. Clusters range from 0 to m-1 where C_k denotes a k^{th} level cluster. $C_k(s)$ is the k^{th} level cluster which contains the node s [32]. The connections between both nodes and clusters are links. For the purpose of measuring the path length between two nodes in the network, each link is counted as unity, the equivalent of one hop count. We illustrate the logical hierarchy in Figure 3.3

While the logical hierarchy can be described as a tree, it does not imply that network data flows on the links of this logical tree.

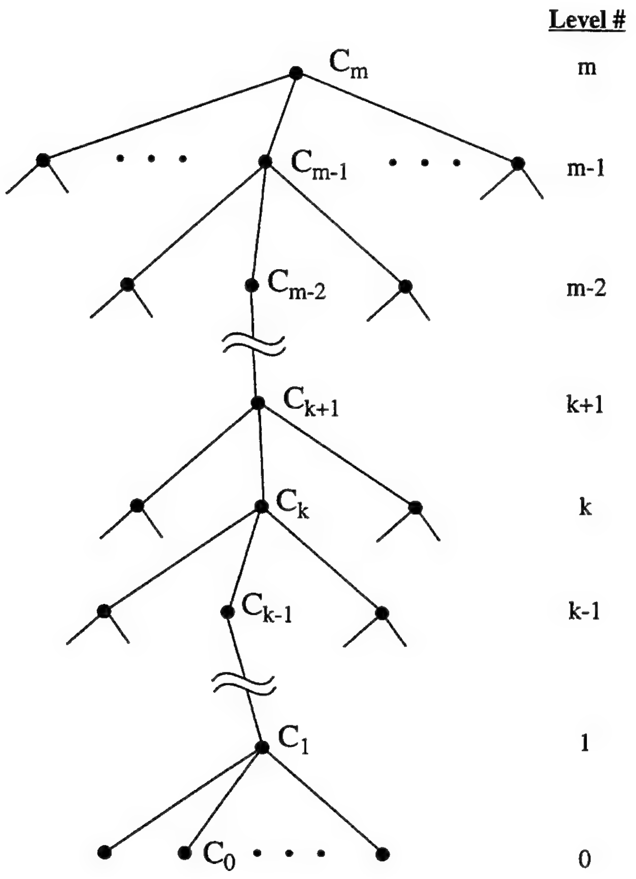


Figure 3.3: Logical Tree Representation of a Clustered Network. From Ref. [21]

a. Level n

Level *n* refers to a level in the logical tree at which the different multicast components exist and operate (i.e. level *n* clusters as defined in Definition 3.7). The prefix, *n*, implies that the components are at the same level with a common parent at level *n*+1. There are several level *n* components which are described individually below.

4. Groups

Groups are a collection of member nodes who either wish to send or receive multicast data as defined in Reference [13]. The members of a group are end systems on a local area network (LAN) and are represented by a designated router (DR) for their LAN. The DR is a level 0 cluster. A group's scope is determined by the expected spread of the locations of the members. The group level is the same as the scope level and refers to the level of the lowest common cluster in which all group members are included. The scope of the group is determined at the time of the group's creation. Groups with different scopes are illustrated in Figure 3.4.

> <u>*Definition 3.9:*</u> A ***level n group*** is a group whose members belong to the same parent cluster C_n and whose scope is level n.

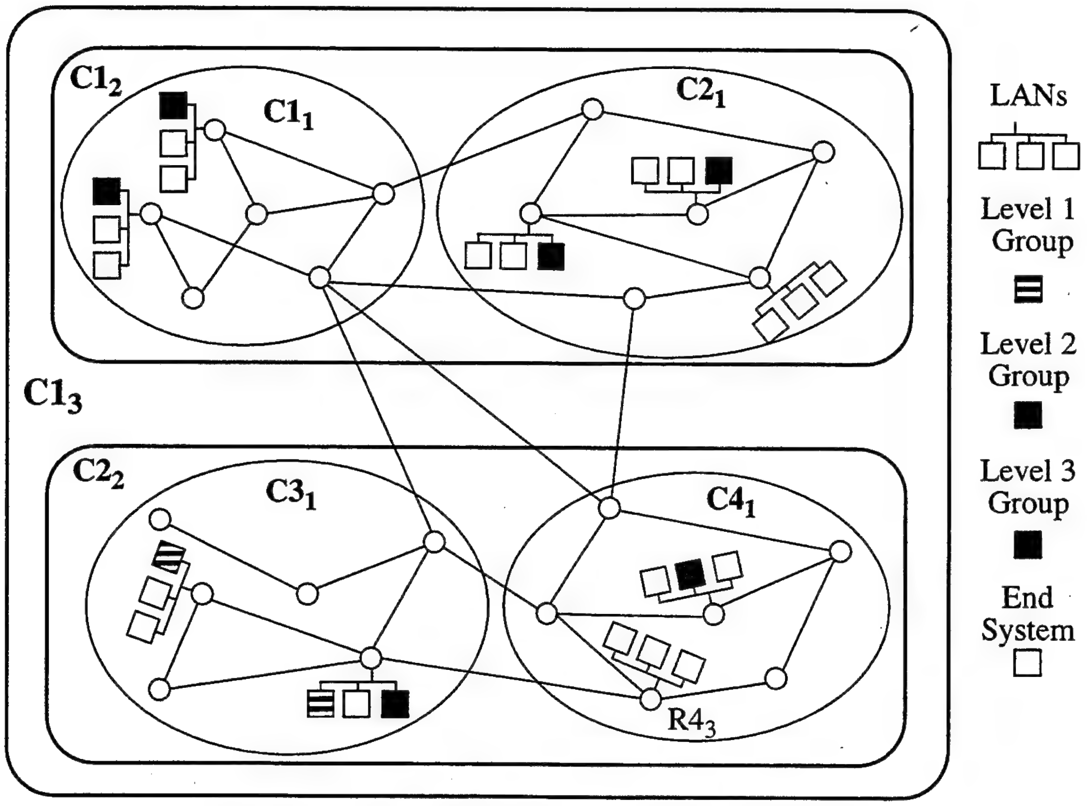


Figure 3.4: Sample Groups in a 3 Level Hierarchy

5. Multicast

Multicast refers to the actual data distribution. This assumes that the data distribution paths have been set up for proper routing of the traffic. A multicast-capable router is one with the ability to perform multicast operations.

> *Definition 3.10:* A ***level n multicast*** is a the distribution of data to members of a level n group.

6. Multicast Border Router (MBR)

> *Definition 3.11:* A ***level n multicast border router (MBR)*** is a multicast-capable level n border router where a level n border router is defined in section C.2 above.

A multicast border router, A, at level n is denoted as A_n.

D. ASSUMPTIONS FOR HIERARCHICAL MULTICAST

Before we can explain the details of the multicast hierarchy, additional assumptions are stated regarding the underlying network and the services it provides to multicast. These assumptions are based, in part, on existing or proposed implementations.

The distance and delay information used for the proposed hierarchical multicast is based on the assumption that there exists a unicast routing mechanism which supplies the shortest path between any pair of border routers. Thus:

Assumption 3.6: There exists an underlying unicast routing mechanism based on domains and border routers such that a border router knows the shortest path to any other border router [43].

The above assumption is based on present Internet terminology and implementations. The following is based on proposed implementations:

Assumption 3.7: All Border Routers are Multicast Border Routers (all are multicast-capable)

An interim solution prior to Assumption 3.7 being implemented is the use of *tunnels.* Tunnels encapsulate multicast traffic in a unicast message and use unicast protocols to send data between multicast-capable routers[49].

Note that we also adopt Assumptions 3.3 and 3.5 of the hierarchical unicast case, the clusters are connected and the shortest path between two nodes in a cluster lies within that cluster.

E. CHARM FUNCTIONAL COMPONENTS

In this section, we provide a functional description of the CHARM components. In those cases where it is appropriate, an algorithmic description of the components behavior is described.

1. Clusters

Clusters, defined in section C.2, play specific roles in a given multicast. A cluster can be a sending cluster, receiving cluster, transit cluster, or any combination of the three. Note that group members can be senders only, receivers only, or both.

Definition 3.12: A ***sending or source cluster*** for a group is a cluster with members who send data to the multicast group.

Definition 3.13: A ***receiving cluster*** is a cluster with members who receive multicast data.

Definition 3.14: A ***transit cluster*** is a cluster with no members but lies along the shortest path between two clusters such that multicast data transits through one or more of its border routers.

2. Registrar

When a group is created, it is advertised by an agent responsible for group registration. This agent is known as the registrar. The registrar is responsible for doing group name and address resolution. This function is presently carried out by the session directory(sd) tool [30]and is described by the session directory protocol [25]. The sd tool, however, is not hierarchical. The registrar we propose is level-specific and performs some additional functions.

A registrar at level n is expected to be a distributed replicated directory service for a level. Each cluster at every level has a "local" registrar. The level n registrar consists of the local registrars for the clusters at this level. All the registrars at one level maintain information about groups at that level in a replicated fashion in that the information for level n is found in each of the local cluster registrars at level n. The level n control tree, a shared tree connecting all the registrars for clusters at one level with a common parent, is used to pass the information between local registrars.

We describe the function of the registrar to support the hierarchy in the following ways:

- to advertise groups and associate a group address with a group name *for a particular level,*
- to be a member of the control tree at its level,
- to be a member of the all-MBR tree in its cluster, and
- to store and disseminate, when requested, the MBR information for its cluster.

The information the registrar maintains facilitates group creation, sender registration and receiver join. The registrar is structured such that level n information is kept at level n.

Figure 3.5 shows an example of registrars at a level. The registrars for a level make up the level n control tree (see section E.4). Within each of these clusters, there may be one or more leaf level clusters which also have their own registrar.

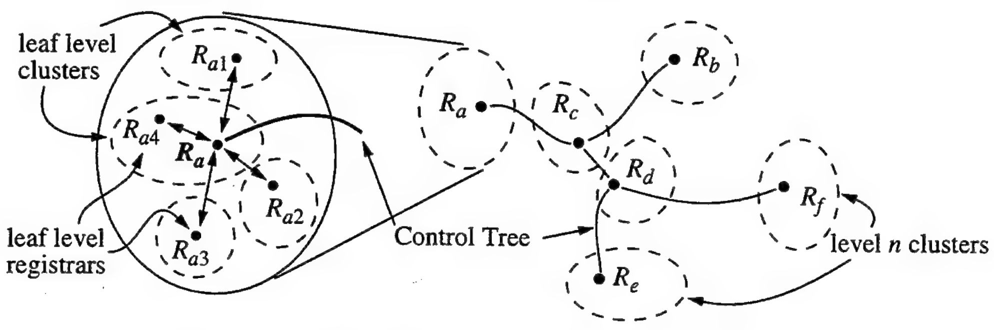


Figure 3.5: Registrar Control Tree for a Level

Leaf level registrars know the addresses of each parent registrar in its part of the hierarchy. Level *n* registrars do *not* need to know the addresses of their leaf level registrars. The registrars at the leaf level communicate with their level *n* parent registrar using unicast routing. Any unicast messages that are used for retrieving level *n* MBR information must include a return address. Each of the registrar functions is described below.

a. Group Creation

A request to create a group is made to the leaf level registrar. The leaf level registrar is available on a well known multicast address within a leaf level cluster. Based on the requested scope of the group, say level *n*, the registrar forwards the group creation request to the local level *n* parent registrar. The level *n* registrar then advertises the group at its level, sharing the existence of this group with all other level *n* registrars.

This process is illustrated in Figure 3.6. The originator's group creation request is sent to the leaf level registrar. The leaf level registrar, R_{a1}, forwards this group creation request, with its own address as a return address, to the level *n* registrar R_a. R_a returns the group address and distributes it to the other registrars at this level, as shown, on the control tree. The other registrars use this to make an entry in their group membership tables which are used when members of this group appear in the leaf level clusters below them.

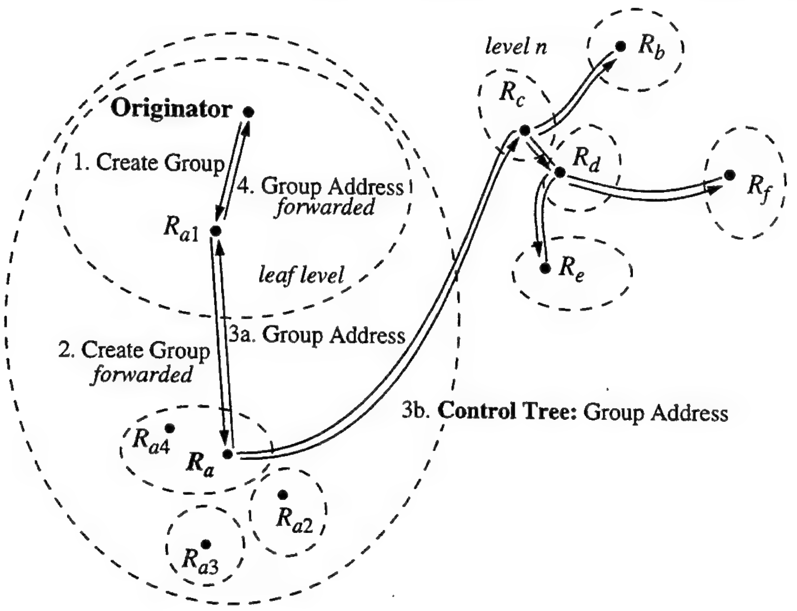


Figure 3.6: Registrar Actions for Group Creation

b. Sender Registration

Senders to a group at any level send the registration request to the closest leaf level MBR. A sender obtains the list of leaf level MBRs from its leaf level registrar. Once a sender obtains a leaf level MBR list, the closest MBR is cached for future use.

The leaf level MBR forwards the registration to its closest level *n* MBR. The level *n* MBRs are available to the leaf level MBR from the leaf level registrar. The leaf level registrar gets the level *n* MBR list from the level *n* registrar, prior to any group creation requests. Level *n* registrars maintain a list of level *n* MBRs, for all *n*, as part of the static configuration of the hierarchy. The leaf level MBR is expected to cache its closest level *n* MBR for each level *n*, for $n > 1, \forall n$.

Once the level *n* MBR receives the new sender registration, it forwards it out on the all-MBR tree, a shared tree specific to each level *n* cluster that connects all level *n* MBRs of that cluster and the local level *n* registrar (see section E.4.b). The level *n* registrar being a member of this tree, distributes the presence of this sender on the control tree, of which also it is a member, in the form of a new source cluster notification message. The new source cluster notification contains the level *n* MBR list from this cluster. It is intended to initiate a Designated Border Router (DBR) election, as described in section E.3, in the other clusters at this level. The registrar need only do this once for all groups in a source cluster, that is, when the first sender for any group in this cluster appears. After a DBR is elected, only the source cluster and group address need to be passed, the DBR is valid until all sources have left and it is explicitly pruned. Each group address stored with a level *n* registrar is tagged with a source bit. After a sender joins a group, the source bit is marked as active. This source bit is reset when the last sender in this cluster leaves, enabling the registrar to initiate a DBR election in other clusters at this level when a sender reappears.

Thus, the leaf level registrar interaction for a sender join accesses a cached MBR entry at a sender, a level *n* registrar accesses a cached MBR entry at a leaf level registrar and a source cluster notification is sent on the control tree.

The sender registration process is shown in Figure 3.7 with the sequence of events identified. It is important to note that once events 1 and 2 have occurred, the information is cached such that they will not occur for subsequent registration from this node.

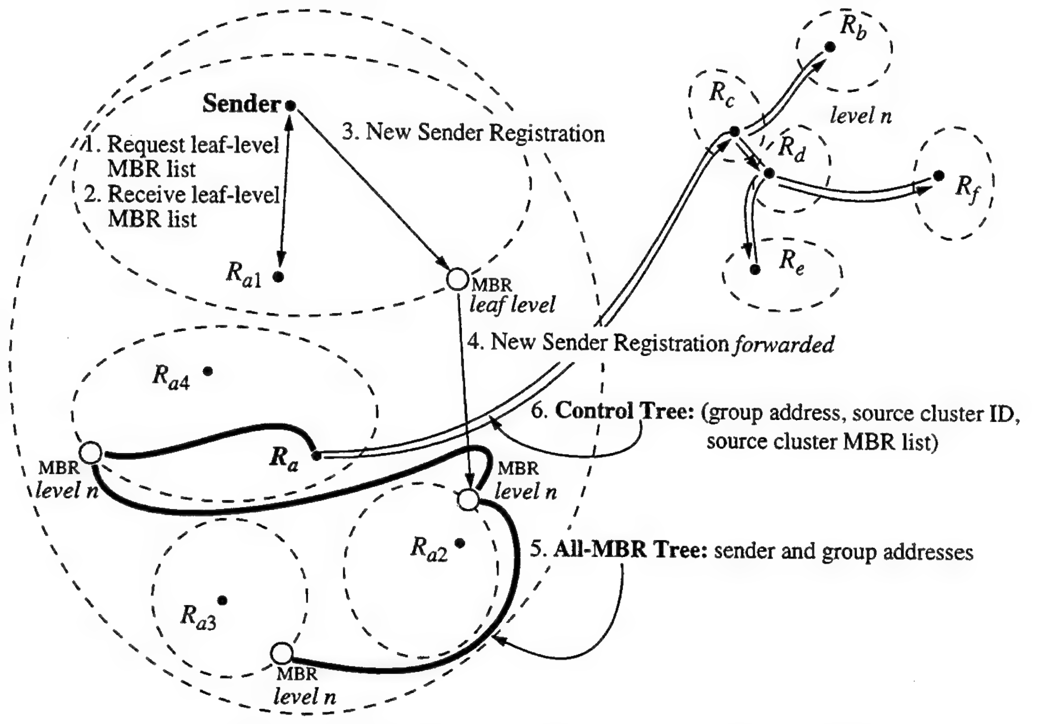

Sender
1. Request leaf-level MBR list
2. Receive leaf-level MBR list
3. New Sender Registration
R_{a1}
MBR leaf level
4. New Sender Registration forwarded
R_{a4}
R_a
MBR level n
6. Control Tree: (group address, source cluster ID, source cluster MBR list)
MBR level n
R_{a2}
5. All-MBR Tree: sender and group addresses
R_{a3}
MBR level n
R_b
R_c
level n
R_d
R_f
R_e

Figure 3.7: Registrar Actions for New Sender Registration

c. *Receiver Join*

A potential group member its leaf level registrar for the existence of a group. The leaf level registrar queries the appropriate level *n* registrar. This information is provided only in response to a request and should not be cached as groups are dynamic.

Level *n* MBRs of a cluster register with their level *n* registrar. They do this when they form the all-MBR tree for a cluster by having the local registrar act as the center of this tree (see section E.4.b). The level *n* registrar maintains a list of its level *n* MBRs, which is static information that changes only when clustering changes or MBRs fail/recover. When a receiver queries about a group's existence from the leaf level registrar, it is provided a list of MBRs for its local level *n* cluster at the level of the group. The receiver uses this information to look up the closest MBR to join for externally sourced traffic.

The information flow for a receiver join is shown in Figure 3.8. Leaf level registrars are also expected to cache MBR entries for levels which have group activity. Thus, steps 2 and 3 do not occur once this information is cached at the leaf level.

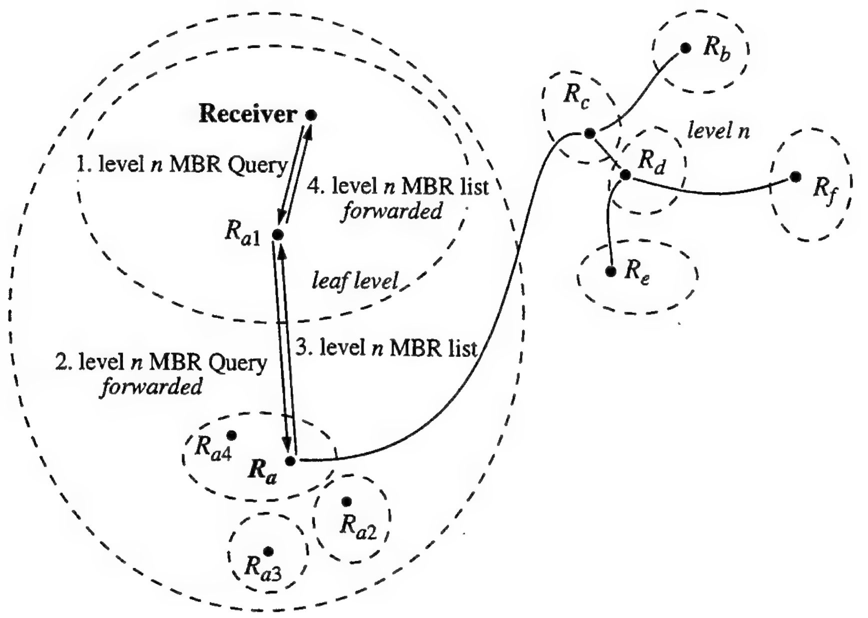


Figure 3.8: Registrar Actions for New Receiver Inquiry

3. Designated Multicast Border Routers

An MBR of a receiving cluster can become a designated MBR (DBR) for a source cluster. A DBR is specified for a receiving cluster as the entry point for multicast data from a given source cluster. Note that the DBR status of an MBR is cluster-specific and not group-specific. A single MBR can act as a DBR for more than one source cluster. The DBR election process is described in Figure 3.9. Ties are broken arbitrarily by selecting the MBR with the lowest network address.

```
    DBR Election at a level n MBR

1   Receive list of source cluster MBRs from the registrar
2   Calculate distances to source cluster(s) MBRs
3   Multicast my distances on my All-MBR tree (see section E.4)

4   If ( my distance is lowest ) then
5       If ( I have receivers attached ) then
6           Join the source cluster MBR
7       else
8           Register source cluster in active (SC,G) table
9       Notify all MBRs in my cluster that I am the DBR for this (SC,G)
```

Figure 3.9: Algorithm for DBR Election

The initiation of a DBR election is in response to a new sender appearing somewhere in the network. If a new sender appears for an existing group and no DBRs have been chosen for this cluster, then the DBR election takes place. This occurs regardless of whether receivers exist for this group. If there are no receivers in a cluster, the (source cluster, group) (SC,G) pair is stored at the DBR and the DBR for this group is stored at all other MBRs until a receiver joins. When the first receiver joins a group for this source cluster by joining its closest MBR, the tree construction begins. If the MBR is not a DBR, it joins the DBR. When the DBR receives a join for this group, signifying the presence of a receiver in its cluster, it joins the source cluster.

The registrar in a source cluster starts the election process by sending a new source cluster notification message on the control tree which contains the source cluster MBRs, the group address, and the source cluster ID. The registrar learns of new senders in the same way that level *n* MBRs do (see Figure 3.7). During the DBR election, each of the MBRs exchanges its shortest distance to some MBR on the source cluster with other MBRs on its ALL-MBR tree. The MBR that elects itself as the DBR forwards out a confirmation on the all-MBR tree. MBRs keep track of the list of DBRs for their groups and the DBRs keep track of the (source cluster, group) (SC,G) pairs. Group information is required since externally sourced multicasts in receiving clusters uses (SC,G) entries in their routing tables because there is no group aggregation, only sender aggregation at the source cluster.

4. Control Trees

There are two types of control trees in the hierarchy. The all-MBR tree connects all MBRs and the local registrar of a cluster for a specific level. There are multiple all-MBR groups in the network, one for each cluster at a level. The second type is a level n control tree which connects all of the registrars for level n with the same parent cluster at level $n+1$. The all-MBR trees are used to pass low frequency cluster-specific information about MBRs and group-specific information. The control trees are used to pass level-specific information between registrars. We show these trees constructed for a sample network which we will use later to describe the operation of CHARM.

a. Level-specific Information

All registrars are members of the control tree for their level. This tree is formed prior to any multicast groups are formed. The tree is a center-specific tree formed using a well known multicast address. The center for this tree is selected administratively.

The control tree has two primary purposes. It is used in support of the distributed replicated group directory service which the registrars provide as described in section E.2. It is also used to initiate a DBR election by way of a source cluster notification which includes the MBR list for a new source cluster sent to the other registrars at a level.

A sample 3-level hierarchy is shown in Figure 3.10. The leaf level, or level 1 registrars are members of the level 1 control tree. Two level 1 control trees are shown since the clusters contain two leaf level clusters in each level 2 cluster. A single level 2 control tree connects the level 2 registrars. Note that there is no level 3 registrar at the top of the hierarchy.

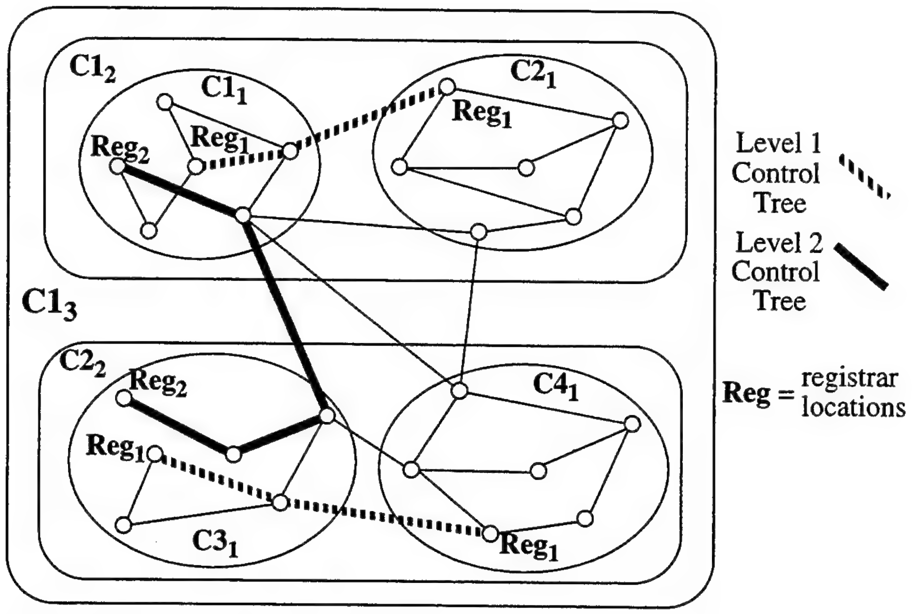


Figure 3.10: Control Trees for a 3 Level, 4 Cluster Hierarchy

b. Cluster-specific Information

The all-MBR tree is critical to certain functions that must be carried out by MBRs. It is also a center-specific tree constructed using a well known multicast address known as the all-MBR group address. The center of the all-MBR tree is located at the node which serves as the registrar of the cluster. The MBRs register with the registrar which is also a center for the all-MBR tree.

The All-MBR tree serves four main functions:

- Senders in a cluster send their registration message to an MBR on this tree (including their first data packet, in case a sender want to send to a group without becoming a member) for all MBRs to join to.
- New source cluster notifications, which include MBR lists, are announced on this tree to initiate the DBR election.
- MBR-source cluster distances as part of the DBR election and the winner of the DBR election are passed between MBRs on this tree.
- It is used to pass periodic refresh and hand-off information within a cluster

between MBRs (see Chapter VI, section B).

MBRs maintain the information regarding local and external senders in an active state table. This state refers to a (source, group), *i.e.* (*S,G*), pair or a (source cluster, group), *i.e.* (*SC,G*), pair which presently exists in an MBR routing table. An *active state* refers to those sources and source clusters which are maintained in the *active tables*. The active tables represent active senders and source clusters that are stored in the event a local receiver should join to the MBR. One of the MBRs learns about a new sender from a new sender notification unicast to it by the sender. This MBR forwards this notification on the all-MBR tree for the other MBRs in the cluster.

New source cluster notifications are multicast on the all-MBR tree by the registrar. These include the source cluster MBR list and trigger a DBR election. The results of the individual distances to the source cluster is multicast on the all-MBR tree so that each MBR can determine the winner. The MBRs store the (source cluster, group) (*SC,G*) pair information in their active tables until joined by a local receiver or until the source cluster DBR send s a leave message.

A 3 level hierarchy is shown in Figure 3.11 with the all-MBR trees. The registrars are also members of the all-MBR trees at their level. We show four level 1 all-MBR trees which connect level 1 MBRs and two level 2 all-MBR trees.

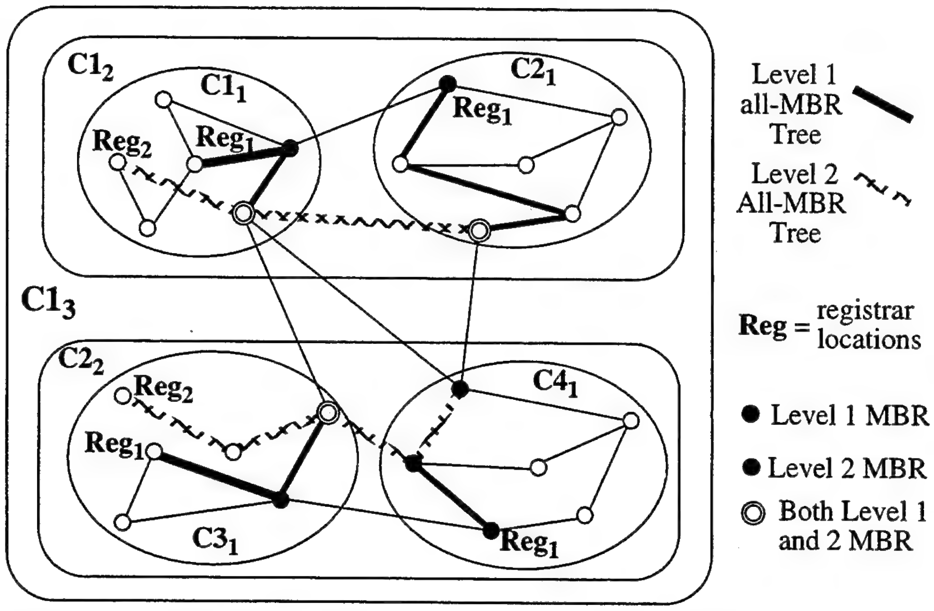


Figure 3.11: All-MBR Trees for a 3 Level, 4 Cluster Hierarchy

5. MBR Functions

Border routers play a special role in both unicast and multicast. For unicast, the purpose behind having border routers stems from the assumption of a "core" backbone of the Internet and that various domains were attached to this core usually with a single router[41]. The special role they play is explained in References [43], [44] and [45] which discuss the inter-domain routing protocols designed for exchanging reachability information between border routers.

The function of the border router is extended for use in a hierarchy. As stated in section C.2, a routing domain corresponds to a leaf level cluster. This aligns all leaf level cluster boundaries with an existing unicast boundary. It is assumed that the unicast border routers perform more functions than an internal router because of the additional requirements that they must perform for unicast, such as acting as gateways to other domains running border gateway protocols [50].

MBR functions in the hierarchy depend on the type of cluster they are in. We now explain these functions for a sending and receiving cluster.

a. Sending Cluster MBR Functions

The MBRs in the sending cluster are responsible for specific internal and external functions. The external responsibilities are primarily to forward data out to other clusters at the same level. The internal responsibilities include joining a group based on a unicast new sender registration or information received on the all-MBR tree. A sender sends a registration request, with its optional first data packet, to its leaf level MBR. The leaf level MBR forwards the packet to its closest MBR at the level of the multicast. This level *n* MBR forwards the multicast on the all-MBR tree and joins the sender. All MBRs who are a point of attachment for a DBR in another cluster at this level for this group, also join the sender. The result is a source-specific tree rooted at each sender to at least one level *n* MBR, with others joining as required. The MBRs inform attached receivers of the new sender on their incoming (source cluster, group) (SC, G) self entry trees. The internal receivers of this group then join the sender.

Figure 3.12 shows the functions of a source cluster MBR which deals with functions of an MBR in a cluster with internal senders.

Source Cluster MBR

```
If ( new sender notification arrives ) then
    If ( unicast ) then
        Multicast new sender notification out on all-MBR tree
        Join the sender
    else
        Register Sender in active (S,G) table
    If ( this MBR has local receivers attached ) then
        Send new sender notification message to local receivers
    If ( this MBR has DBR(s) attached for group ) then
        Join the sender (if not already joined)
    Set state on external interface to forward all traffic for group

If ( sender leave notification arrives ) then
    If ( unicast ) then
        Multicast sender leave notification out on all-MBR tree
    If ( this MBR has DBR(s) attached for group ) then
        If ( this is the last sender for group ) then
            Send a prune downstream along the group's interface
    Un-register this sender from the active (S,G) table

If ( local receiver join message arrives ) then
    Set state on internal interface to self entry for incoming tree
    Forward senders for this group from my active (S,G) table

If ( DBR join message arrives ) then
    Set state on external interface for this group
    Join senders for this group from my active (S,G) table

If ( DBR leave arrives ) then
    Remove state from interface
    Send leave message to internal sources
```

Figure 3.12: Functions of a Source Cluster MBR

b. Receiving Cluster MBR Functions

The multicast border routers in the receiving cluster are responsible for forwarding data to local receivers from external source clusters. Figure 3.13 shows the functions specifically required by an MBR for external senders.

```
Receiving Cluster MBR

If ( join message arrives from local receiver ) then
    If ( there are DBRs for this group ) then
        Join the DBRs for the group requested
        Set state on internal interface for this group

    If ( I am DBR ) and ( this group not joined to source cluster ) then
        Join source cluster MBR
        Set state on internal interface for this group

If ( leave message arrives from local receiver ) then
    If ( no more receivers attached ) then
        Remove state from internal interface
        Send leave message to DBR

    If ( I am DBR ) and ( this is the last receiver in this group ) then
        Remove state for this group
        Send leave message to source cluster for this group
        Set state in active table source cluster, group pair
```

Figure 3.13: Functions of a Receiving Cluster MBR

F. AN EXAMPLE OF GROUP OPERATION

In this section, we provide an example of a multicast using CHARM. We start with a topology and a group and examine the steps taken to send to receivers in various locations to bring out the features of CHARM.

1. Assumed Infrastructure

Prior to embarking on an example of a multicast group operation, we make the following assumptions about the existence of certain entities and support structures.

We assume that:

- there is a level *n* registrar working at every level as described above,
- the control trees and all-MBR trees have been set up,
- the MBRs are in place and registered,
- we have a group membership protocol running at the LAN level, similar to IGMP [20], and
- end-users know how to get the group name and the level.

We start with a three level hierarchy on a random topology with 6 nodes in a leaf level cluster. The topology is shown in Figure 3.14. In addition, we add one sender and four receivers in a global group at level 3. This same topology was used to illustrate the control and all-MBR trees in Figures 3.10 and 3.11.

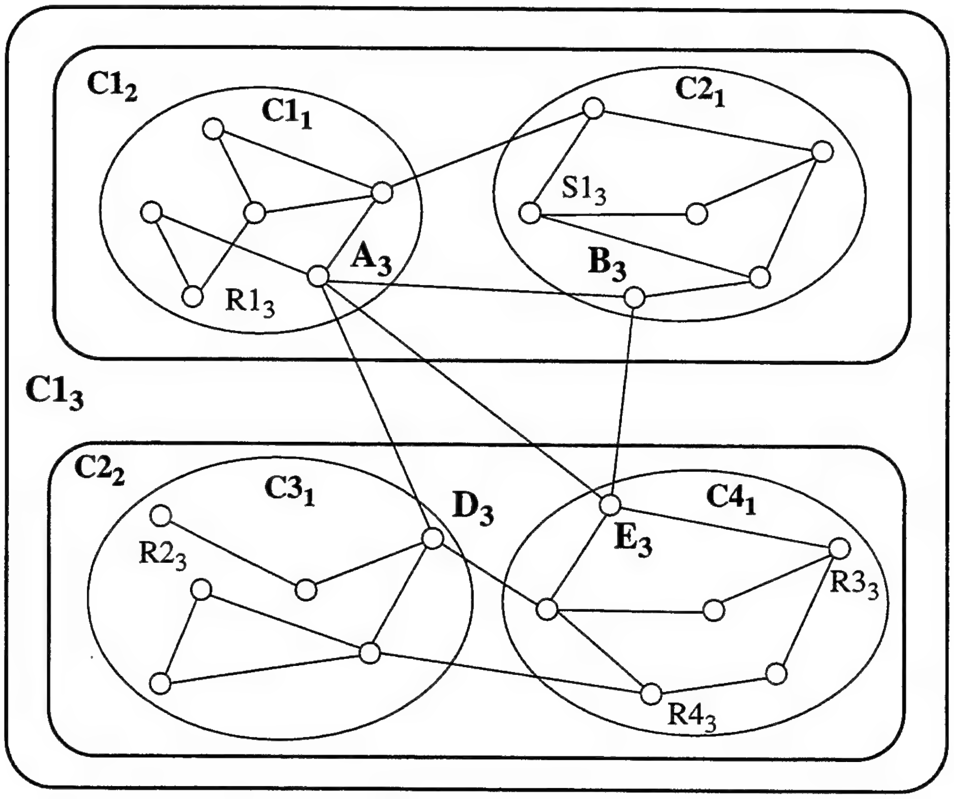


Figure 3.14: A 3 Level Hierarchy with 4 Leaf Level Clusters

It is illustrated here that the diameters of the clusters are related to the path length performance of a multicast. The cluster diameters for this example are in Table 3.2.

Table 3.2: Diameters of the Sample Graph

Cluster	Hops	Level
$C1_1$	3	1
$C2_1$	3	
$C3_1$	4	
$C4_1$	2	
$C1_2$	6	2
$C2_2$	5	
$C1_3$	8	3

2. Creating a Group

The group originator's attached router is responsible for registering the group address with the registrar in its leaf level cluster. The scope of the group is determined at creation by the originator. The leaf level registrar is responsible for forwarding the request to the appropriate level registrar. At this point, the group has no membership, it is scoped, and only the local registrar and the registrar at the scope level of the group have information about the group.

3. Sender Join

a. New group

For a sender to join a group, once it has received the information from the leaf level registrar about the group address and the leaf level MBRs, it sends a new sender registration to its closest leaf level MBR. The leaf level MBR forwards the registration to its closest level *n* MBR with the sender's address and the group address. The level *n* MBR forwards the sender information out on the all-MBR tree and then sends its own join to the sender. Each of the other MBRs in this cluster join the sender if they have receivers attached. Otherwise, they store the (source, group) (S,G) pair in their active source table. Since this is a new group, no receivers are present. The sender can continue to send data to its closest leaf level MBR until it receives the join request from the level n MBR which received the unicast registration, at which time, it stops sending to the leaf level MBR and multicasts directly on the newly formed tree.

This result is shown in Figure 3.15. We drop the subscript 3 for now since the only group we have is level 3. MBR B received the unicast new sender from the leaf level MBR and multicasts on the all-MBR tree which includes MBR A. MBR A stores the (S,G) pair information in its active table and B joins the sender.

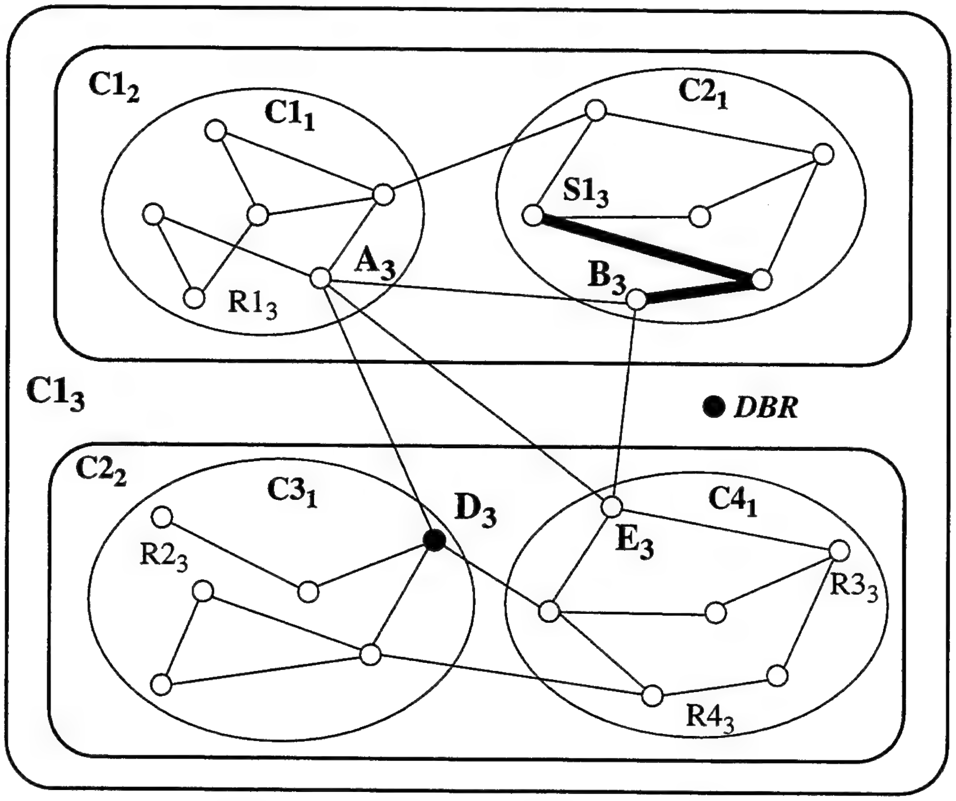


Figure 3.15: Sender $S1_3$ Joins the Group

b. Existing Group

If there are existing members in a cluster for a group, then local cluster receivers need to be notified of a new sender. When the sender unicasts its registration to its closest leaf level MBR, and then to the level n MBR, the registration and data flow initially on the all-MBR tree. The registrar, also a member of the all-MBR tree, sends a new source cluster notification on the control tree. This message which contains the MBRs from the source cluster, is multicast on the all-MBR tree of the receiving cluster and a DBR election process takes place.

MBRs receive the sender's information and data and join the new sender if they have receivers already attached. If there are receivers for this group in this cluster, they are joined the closest MBR. The initial join by a receiver constructs a branch from the MBR to the receiver using a self-entry for a (source cluster, group) (SC,G) pair router table entry. The MBR uses this entry to forward new sender information to all receivers joined to it along this tree. The receivers then join the new sender directly.

In our example, when the sender S1 joined the group, a new source notification gets sent to cluster $C2_2$ with MBRs A and B as part of the message. The receiving cluster registrar in $C2_2$ passes MBRs A and B to D and E on its all-MBR tree. Both D and E have a hop count of 1 and D is chosen because its address is lower. No join ensues since no receivers exist yet in $C2_2$, but the (source cluster, group) (SC,G) pair are stored in the active source cluster table at D and the DBR, group pair is stored at E.

4. Receiver Joins

Using a receiver-initiated join model, the receiver's join request begins the tree construction process. Using either existing explicit join methods, CBT or PIM, a receiver propagates a join request towards a sender and its nearest level n MBR in the local cluster.

The receiver initiates a level n MBR query prior to joining the group. R1's query, in this case, returns the MBRs A and B. R1 chooses A as its closest MBR and sends a join for this group to A. MBR A constructs a branch back to R1 using, for example, a CBT-like method where a join acknowledge is sent back to the requesting node which creates the tree branch [1]. The state information for this branch is the self entry (SC,G) pair. A also sends on this branch the senders for this group from its active (S,G) table. R1 can now join a sender directly.

We continue with the example by showing the tree constructed when R1 joins the sender at S1. Thus MBR B and R1 are on a single SST rooted at S1. The resulting trees are shown in Figure 3.16.

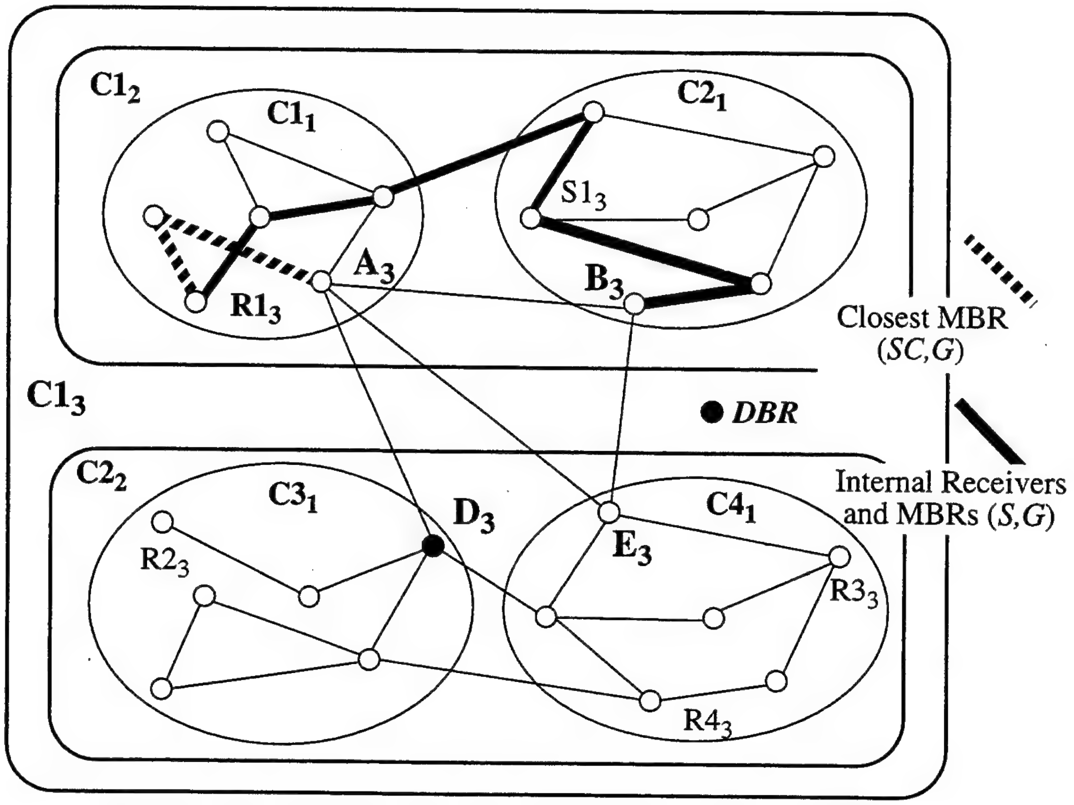


Figure 3.16: Receiver $R1_3$ Joins the Sender and Closest MBR

Next, we illustrate how receivers in cluster $C2_2$ join the group. The leaf level registrars in $C3_1$ and $C4_1$ return the MBRs D and E to each of the receivers, R2, R3, and R4. R2 joins first and chooses D as its closest MBR. D constructs a self-entry branch out to R2 and then based on its active (SC,G) table, also acknowledges the active source cluster for this group and adds an (SC,G) branch for the source cluster $C1_2$. D is now ready to send a join to A for this group. A, having an (S,G) entry in its table for the group, is able to know to join the sender at S1, grafting a branch on the existing tree constructed by R1.

The resulting tree is illustrated in Figure 3.17. In this example, R2 gets a shortest path tree to this source. No receiving cluster penalty is incurred because the MBR R2 joined is also the DBR for this source cluster.

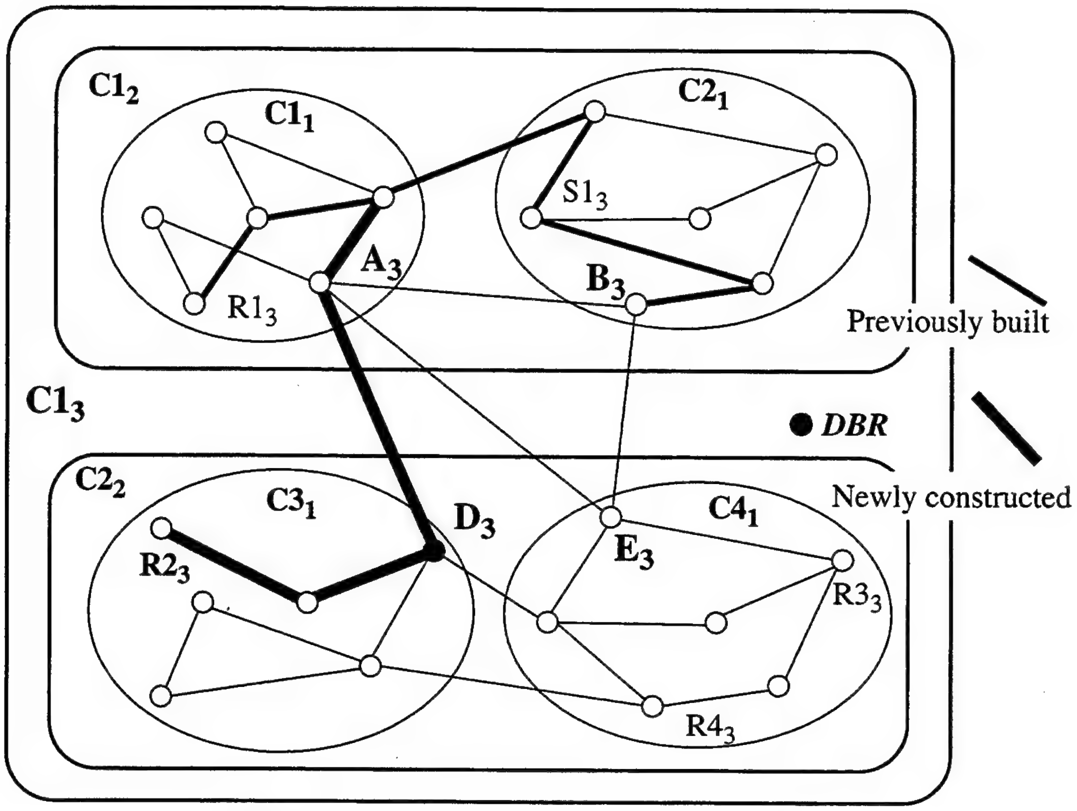


Figure 3.17: Receiver R2$_3$ Joins the Source

The next case shows when a receiver does not join the DBR for the source cluster. Receiver R3's nearest MBR is E so it joins E. MBR E, having an entry in its tables showing that D is a DBR for this group, then joins D. The resulting tree is shown in Figure 3.18. In this case, the shortest path to the sender without the hierarchy is 4 hops but the path taken is 7 hops. This receiver is incurs a penalty in both the sending and receiving clusters. In this instance, it is a 1 hop penalty in the source cluster and a 2 hop penalty in the receiving cluster.

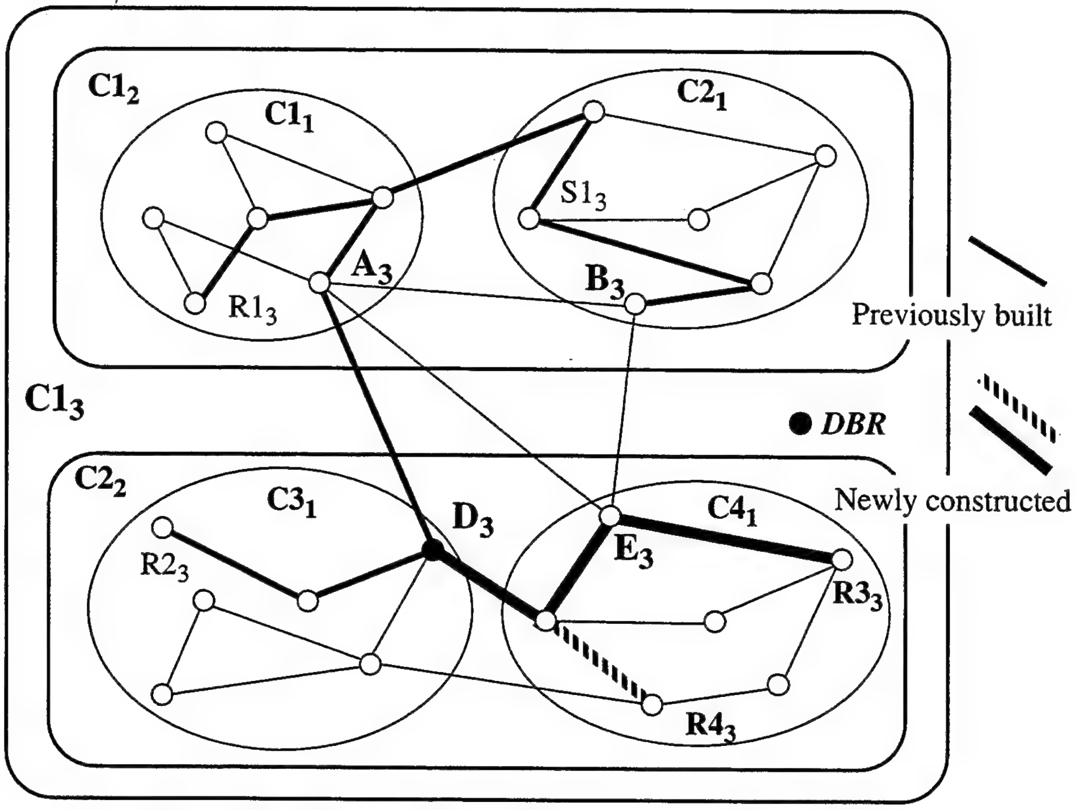


Figure 3.18: Receivers $R3_3$ and $R4_3$ Join their Closest MBRs

The last receiver to join is R4. R4 also chooses E as its closest MBR. However, on the way to joining at E, the branch is grafted. The join message must still propagate to E in case there are other source clusters for this group. This is illustrated in Figure 3.18. In this case, R4 incurs only the source cluster penalty which is only 1 hop more than the shortest path.

a. Join Latency

One drawback of a receiver-initiated join is latency. Often, this issue is related to hard state vs. soft state as discussed in Chapter II, section C.2.b. The receiver-initiated, explicit tree construction method we describe here is produces hard state for all trees which does not time out but rather requires explicit leaves to terminate branches. It should be noted that the join latency for a group is only incurred by the first receiver to join the tree.

Since we allow for source aggregation at the source cluster, this amounts to the first receiver for a group in a receiving cluster.

5. Sender and Receiver Leaves

Join latency and leave latency are related. Implicit leaves are addressed in DVMRP in Reference [20] and they refer to the practice of allowing branches of the delivery tree to time out. The reason that the leave mechanism is important is that if a receiver is allowed to leave a group passively (i.e. time out) then the branch to the receiver continues to use bandwidth unnecessarily for some time after the receiver is gone.

In the hierarchy, groups with the same source clusters are likely to aggregate multicasts on the inter-cluster trees. As a result we require all senders and receivers to explicitly leave their groups by following a procedure similar to the join. Senders are required to send leave messages to their closest leaf-level MBR as for a new sender registration. Local receivers send leave messages directly to the senders.

Source clusters must terminate their existence explicitly since many senders could be in one source cluster. DBRs who have had their source cluster terminate then notify their locally attached receivers.

Receivers send leave requests to the MBR they joined to. If the MBR has had all receivers leave, it sends a leave to the DBR(s) it is joined to. If a DBR has no more receivers or MBRs attached, then it propagates a leave to the source cluster. In this fashion, no extraneous branches are left after all group members have terminated their existence in a cluster.

G. ILLUSTRATION OF SCOPE CONTROL

Using the same sample network, we examine a level 2 multicast. In this case, the sender is in $C3_1$, and the receivers are in $C3_1$ and $C4_1$. C is chosen as the DBR.

The MBRs are different for this level, except for B_2 which also acts as an MBR at level 3. This shows how for multicasting at different levels of the hierarchy, the data does not follow the logical hierarchy. It is not required for data to go to a level 2 MBR prior to going to a level 3 MBR.

The final result of the tree construction is shown in Figure 3.19. R1 joins S1 directly. Both R2 and R3 join the DBR at C even though R2 is an MBR. In this case the penalty incurred by R2 is in both the sending and receiving cluster. R3 incurs only a sending cluster penalty of 1 hop. The difference between this and levels above is that with scope control and clustering, the penalty is reduced because the diameter of the clusters is reduced.

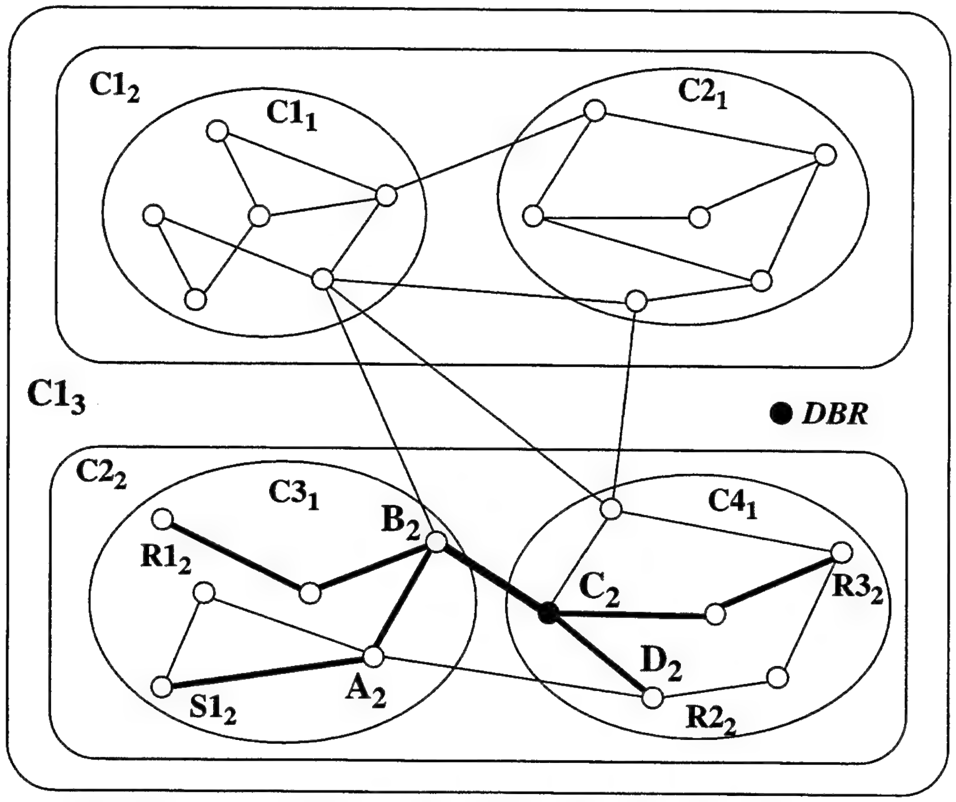


Figure 3.19: A Level 2 Group where R1_2, R2_2, R3_2 Join S1_2

H. PERFORMANCE RELATED ASPECTS

1. Aggregation

CHARM allows for source aggregation. If a source cluster has many senders, they are treated as one sender outside of the source cluster. This is a savings in routing table entries over the source-specific tree which requires individual sender information to be maintained with the group address for a multicast. This is not as good as a center-specific tree which, in the absence of group aggregation, achieves the best savings since all senders are aggregated on the tree and no sender information is required [1].

2. Path Length

We have pointed out in the examples in this chapter the potential for increased path lengths due to the hierarchy. We have shown that the path length penalty is dependent on the sender and receiver locations with respect to the MBRs and the DBR election process. The path length penalty is discussed in detail in the next chapter.

3. State Space

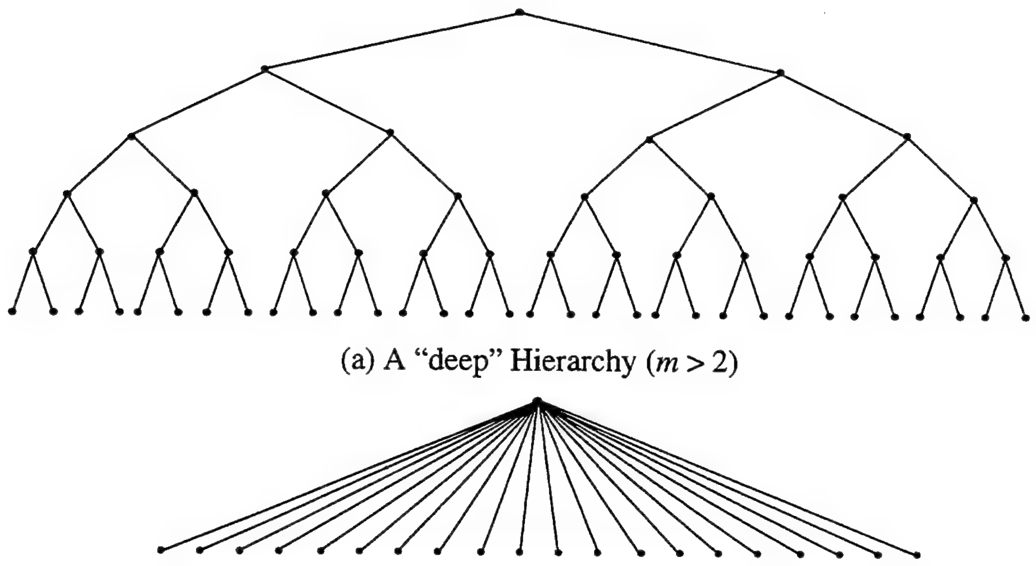
State space is the defined as the amount of information required by a router to support the multicast. Large amounts of this information translates into storage requirements and potential performance penalties. Much of the processing for routing consists of table look-ups, which can be efficient if the table sizes are kept small. CHARM addresses this in two ways. First, the storage is saved through source aggregation. Second, through scope control, fewer nodes participate in the routing for a multicast.

State space requirements for the source-specific trees in a source cluster can be high if there are many senders in the source cluster. This could be relieved by using a center-specific tree in the source cluster with the concomitant path length penalty. State space is considered in greater detail in Chapter VI on network overhead.

4. Scope Control

One of the primary benefits of a hierarchy for multicast is scope control. Granularity of the scope control possible is related to the number of levels of the hierarchy. The deeper the hierarchy is the finer the control over the scope of the group.

The illustration in Figure 3.20 shows the scope control gained by having a "deep" hierarchy vs. a "flat" one. In a two level hierarchy, any multicast that is not local is global.



(a) A "deep" Hierarchy ($m > 2$)

(b) A "flat" Hierarchy ($m = 2$)

Figure 3.20: A Flat versus Deep Hierarchy

IV. PATH LENGTH PERFORMANCE

A. GENERAL REQUIREMENTS

A primary performance-related goal of the proposed multicast technique is to have the data propagate along the shortest path from any sender to any receiver in the group. The path length directly affects the delay as seen by the receiver. This end-to-end delay measurement is our primary metric for evaluation of CHARM.

In analyzing multicast performance, delay is a difficult measurement to quantify because it is composed of several other components. They are processing delay, queueing delay, transmission delay and propagation delay [5]. Each represents a behavior of the network which can be modeled individually, however, models which incorporate all become intractable for a large scale network. Path length is directly related to propagation delay in that it translates to a geographical distance. Path length, measured in hop counts, can also be related to processing and queueing delay since each hop represents some processing and queueing at the intermediate nodes. A factor to be considered in modeling a complex network is that the link delays are dynamic based on load. In our work, we do not concern ourselves with changing network loads and with the overall cost of the tree. Thus, we characterize the performance of CHARM in terms of the delay between senders and receivers of a group relative to the delay along shortest unicast paths. We refer to increased delay due to the hierarchy as the path length penalty. By focussing on the penalty, we bound the worst case penalty and show how to improve it.

The objective of this chapter is to spell out alternative approaches on how to best achieve the minimum penalty for a hierarchical multicast. For a cluster-based hierarchy, there are three elements that need to be examined, namely, the source cluster, between clusters and receiving cluster. We show that in CHARM paths, a deviation from the shortest unicast path occurs only inside the source and receiving clusters. The inter-cluster path is insured to be the same as the unicast shortest path by CHARM.

B. INSIDE SOURCE CLUSTERS

We assume that a cluster at any level has multiple MBRs. It is undesirable to force a single exit point out of a source cluster as we do not make any assumption about the connectivity of the clusters at a level, other than that they are connected (see Assumption 3.3). Therefore, if, two inter-cluster paths for distinct receiving clusters and the same source cluster, share a common transit cluster, it is not required that the transit cluster's exit MBR for both the receiving clusters be identical. A single exit point on the transit cluster would mean only one of the two receiving clusters can receive data along the shortest path. Another reason to do multiple exit points is to help minimize the source cluster penalty.

We examine three approaches for getting the data out of a source cluster that affect the path length for the internal receivers, that is, receivers for the group inside the source cluster, differently.

1. Shared Tree Approach

The tree along which a sender sends data to its cluster MBRs and its internal receivers is referred to as its outgoing tree. A simple and straightforward approach is the use of a single shared outgoing tree to which all internal receivers and MBRs of the source cluster would join. This is a low router state overhead solution and it requires the advertisement of a single center for receivers to join to and senders to send towards. It is an efficient use of state space since it is a single outgoing tree per group and only one router table entry per group is required. A problem with this solution is that it incurs a potential path length penalty to the internal receivers. A second problem is that the MBRs, or the exit points, may incur a similar penalty.

The distance along a center-specific tree from the sender to the MBR at the level of the multicast is at most twice the shortest path between two points on this tree [53]. In the worst case, the shortest path is the diameter of the cluster. Thus a worst case penalty of one cluster diameter is potentially incurred for the MBRs and the internal receivers. This penalty to the internal receivers can be eliminated by using source-specific trees rooted at each sender, allowing for internal receivers and MBRs to receive the data along the shortest path. If this is done, for those members of the group internal to the source cluster, no penalty would be incurred.

2. Broadcast to all MBRs

In order to determine which MBRs need to join senders, a determination must be made about which MBRs will forward data out of the cluster. A simple solution is to have all MBRs join all senders. The MBRs then receive data regardless of whether they are exit points and a forwarding decision is all that is required to begin forwarding data out. This leads to a lower join latency for receivers outside the source cluster.

A problem with this approach to outgoing trees is a waste of bandwidth and state space. If there is no need to forward data out of a cluster and the MBR itself is not a member, then having the MBR join is wasteful. It is particularly so in a source cluster with a large number of senders. An explicit local join, as necessary, is a better method.

3. Explicit Local Joins

The decision on whether or not to forward data out of a particular MBR of a source cluster needs to be based solely on the presence of receivers in an external cluster that request receipt of data out of that MBR. The decision if an MBR needs to forward data out is made outside of the source cluster. This method is in keeping with the receiver initiated join principle. Once an MBR receives a join from a DBR, it joins all senders in its cluster for that group. This is a more efficient use of state space and bandwidth.

Thus, CHARM requires that internal receivers join local senders' source-specific trees. MBRs join the source specific tree only if a join request is received from outside the source cluster. Level *n* MBRs maintain local sender information in their active (source, group) tables (see Chapter III, section E.4.b). Receivers get the list of senders from their closest level *n* MBR when they join.

C. BETWEEN CLUSTERS

The goal for the inter-cluster tree is to guarantee a shortest path between the receiving and source clusters. The primary reason for this is the unknown nature of the connectivity between two clusters in a large internetwork. The only assumption about the inter-cluster topology is that the end-points are MBRs and, given our definition of leaf-level clusters, they are domain border routers (see section C.2 in Chapter III).

We examine three approaches for getting the data between a source cluster MBR and a receiving cluster MBR along the shortest path.

1. Flood and Prune Approach

A simple solution to insuring that the data arrives at the receiving cluster's MBRs along the shortest path is to use a flood and prune technique. This works well because the clusters have multiple MBRs and this insures that the data reaches all of them along the shortest path back to the source cluster's MBRs. This also assumes that all MBRs of the source cluster forward all data on all external links. This in turn assumes that all source cluster MBRs are joined to every sender inside the source cluster as described in section B.2 above.

This is similar in concept to the approach used in Reference [42] to insure reliable delivery of time-critical messages. The fact that the destination cluster may receive multiple copies works well for receivers joined to their nearest MBR. No DBR would be necessary. In addition, prune messages could be used to reduce the amount of traffic once the shortest paths are known.

The problems with this approach are that the potential for flooding on a global scale is unacceptable for global-level multicasts. In addition, the concept of broadcasting to all MBRs of a source cluster is not acceptable for reasons stated in section C.2. As chosen in PIM and CBT, explicitly constructed paths eliminate these problems.

2. Using Cross-cluster Trees

To build an explicitly constructed source-specific tree, rooted at a source cluster MBR, we need an MBR in the receiving cluster to act as the DBR for that receiving cluster. The role of the DBR would be to receive multicast data on a per source cluster basis and either inject it into the receiving cluster, forward it across the cluster on a cross-cluster tree, or both. A cross-cluster tree is constructed by having all MBRs of a receiving cluster join a DBR and form a source-specific tree rooted at the DBR. A sample network is shown in Figure 4.1 which illustrates the construction of inter-cluster and cross-cluster trees. The resulting tree for a single receiving cluster is a point to point shortest path link from the source cluster MBR, A in Figure 4.1, to the receiving cluster DBR and a source specific tree rooted at the DBR to all MBRs of the receiving cluster. All MBRs in a receiving cluster should be on the shortest path back to the MBR of the source cluster.

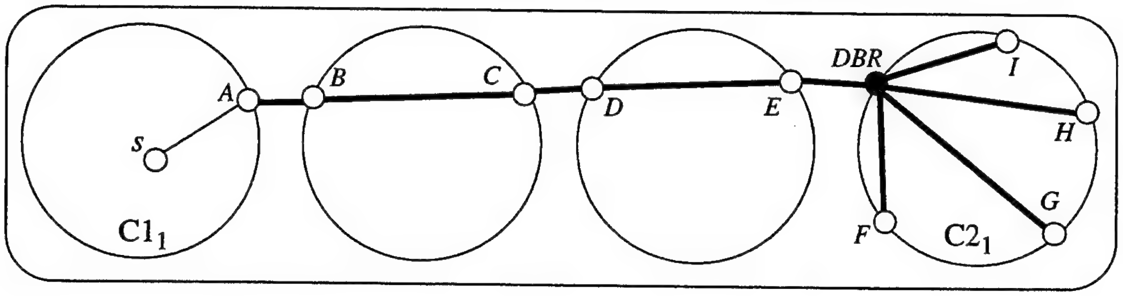


Figure 4.1: Inter-cluster and Cross-cluster Trees

One of the problems with this method is in the construction of the inter-cluster tree. When a join message propagates back to a source cluster, it may traverse other clusters, as in Figure 4.1 above. If the join encounters a cross-cluster tree in another cluster for the desired source cluster, it is forced to graft at that point to the cross-cluster tree. This problem is illustrated below.

For the purposes of illustration, we describe a path as a series of routers. Actually, it is an alternating sequence of links and routers but can be specified by the routers alone since a link is defined by its end-points, which are routers [23]. To further simplify the description of inter-cluster and cross-cluster paths, we shorten the sequence to include only MBRs.

Assume the following relationships for Figure 4.2:

$C2_1$	$\{A, Q\} \leq \{B, D\}$
$C3_1$	$\{B, D, F, G\} \leq \{A, Q, E, H\}$ $\{B, D, F, G\} \leq \{A, Q, F, G\}$ $\{D, F\} \leq diam(C2)$
$C4_1$	$\{H, I, K\} \leq \{G, J, M\}$ $\{H, I, K\} \leq \{G, I, K\}$ $\{H, I\} \leq \{G, J\} \leq diam(C3)$

The DBR for C2, MBR *Q*, joins source cluster C1 at *A*. The cross-cluster tree is formed from MBRs *E* and *F* to *Q* as shown. The DBR for C3, MBR *G*, sends a join for source cluster C1 along its shortest path {*G,F,D,B*}. The join message encounters C2's cross-cluster tree for the same source cluster at MBR *F* and grafts. The worst case penalty due to this graft could be equal to the diameter of C2 (*diam(*C2*)*). The cross-cluster tree for C3 is then formed from *I* and *J* to *G*. Similarly, the DBR for C4, MBR *K*, sends a join on what would be its shortest path to C1 {*A,Q,E,H,I,K*} but encounters the cross-cluster tree of C3 and grafts at MBR *I*. Again the worst case penalty for this graft could be *diam*(C3). This would result in an inter-cluster penalty of *diam*(C2) + *diam*(C3) to every receiver in cluster C4.

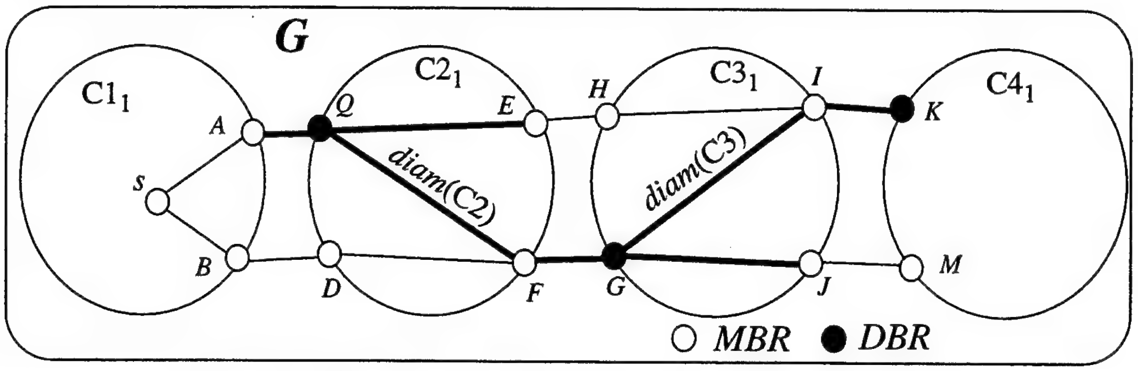


Figure 4.2: The Penalties of the Cross-cluster Tree Approach

Thus, with cross-cluster trees there is a potential for a diameter penalty for every cluster transited. This is unacceptable. The solution is the elimination of the cross-cluster tree and not forcing a single entry point for a cluster.

3. Designated Border Routers

With cross-cluster trees eliminated, explicit joins result in shortest path source-specific inter-cluster trees. The result is a forest formed for each source cluster with the number of trees based on how many different MBRs of a source cluster get attached to. Join requests to a source cluster are allowed to graft to existing inter-cluster trees. The graft would occur to another shortest path tree back to a source cluster MBR, therefore, it would not incur any penalty. As a result, we insure that a receiving cluster has a shortest path to the source cluster.

In the example of Figure 4.2, the DBR of C3 at *G* would not graft but rather construct its own tree back to the source cluster MBR at *B*. The DBR for C4 at *K* would also not graft at *I* in C3 but would graft at *Q* in C2. Thus, each cluster would receive its inter-cluster traffic along a shortest path.

D. INSIDE RECEIVING CLUSTERS

As in the source cluster case, we can have multiple entry points into a cluster, as a result of each receiving DBR building a shortest path back to a source cluster. Initially, we have a DBR per source cluster, which is a single entry point for a receiving cluster. This allows us to aggregate senders on a per source cluster basis for a group.

Receivers do not know who the senders are and should not have to. This allows for open groups as described in section A.2 of Chapter II. It also allows the aggregation of senders in a source cluster. The simplest solution is to give the receivers one location to go to for their data. We need to examine the cost in path length that such a simple solution would incur.

We examine three approaches for getting the data to the receivers of the receiving cluster from an external source cluster.

1. Join the DBR Only

One approach is to have the receivers join towards the DBR. This option is the best choice for receivers with respect to delay because the receiver would be on a shortest path back to the source cluster.

The problem with this is that there may be many DBRs that a receiver has to join, resulting in a potentially large number of joins. This also violates our assumption that the receiver knows little about the make-up of the group. It requires knowledge on the part of the receiver about the location of senders. The complexity of maintaining multiple joins is complicated in the case of DBR hand-offs which will be discussed in the next chapter.

This solution is undesirable due to the amount of knowledge about the group constitution a receiver has to maintain.

2. Join the Closest MBR

A solution which maintains the simplicity of a single join for the receiver is to have the receiver join its closest MBR.The closest MBR is determined after receiving a list of MBRs from the registrar as discussed in the previous chapter. This results in a trade-off between simplicity and performance since the closest MBR may not be on the shortest path back to a source cluster. We examine the penalty that is incurred by this solution in section E on page 71.

An added benefit of this solution is it makes it possible to implement a receiver initiated join in the cluster-based hierarchy. An MBR must receive a join message prior to joining the appropriate DBRs. This amounts to a reduction in the DBR's state space. Only routing information to those MBRs with receivers attached and any receivers that may have joined the DBR directly must be maintained.

3. Join and Graft

A variation on the above approach allows for an improvement to the path length penalty using the features of an existing protocol [2]. If a receiver's join encounters an existing tree between its closest MBR and a DBR, it grafts onto the tree at that point. The join message must still propagate to the MBR in case there is more than one DBR for this group and to provide a self entry branch for propagation of new sender notification messages. The example in Chapter III in Figure 3.18 illustrates this concept when R4 joins.

One other type of graft is permitted. We allow MBRs with receivers attached who find themselves on another cluster DBR's shortest path back to a source cluster to graft onto that tree. This is only allowed if the entry for this tree matches the both the source cluster and group. If the tree is for another group then a graft is not done, since technically it is a different tree. But now that the MBR knows that it has a shorter path back to a source cluster, it is allowed to propagate a join request along this path. The MBR continues to receive data from its own DBR for the source cluster, group pair until this new branch/tree is constructed, after which it prunes itself for this source cluster, group pair from the DBR. This will only occur if the MBR has already joined its own DBR, if it has not, then the prune will be unnecessary.

We use the sample graph in Figure 4.3 to illustrate this point. The receiver, *r*, joins to its closest MBR at *F*. *F* is on the shortest path for the same group and source cluster along $\{B,D,F,G\}$. Thus, *F* grafts to that tree and does not join the DBR for this source cluster. If there were other source cluster DBRs, the normal join process would continue.,

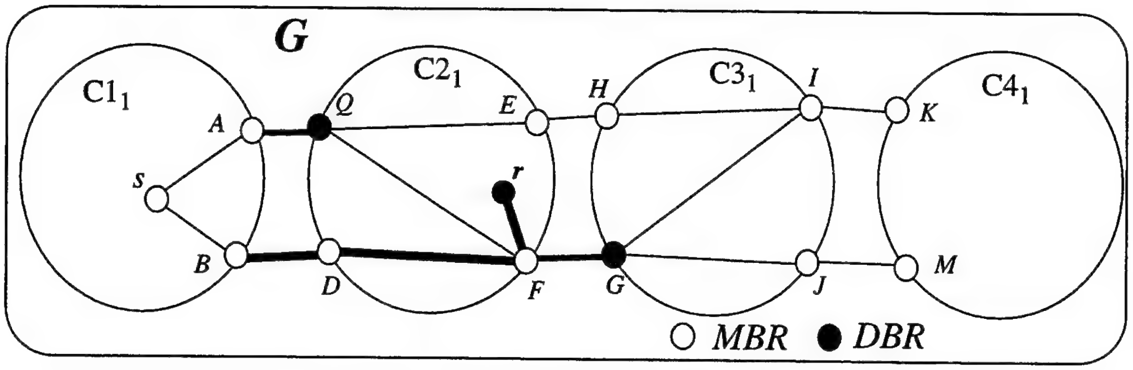


Figure 4.3: Grafting to an Existing Tree

E. AN UPPER BOUND

We have now examined the components of a path in CHARM that make up a complete path from any sender to any receiver in the network. Receivers inside a source cluster incur no penalty since they use source-specific trees to every internal sender. Inter-cluster trees are source-specific trees and as such, no penalty exists between MBRs of source and receiving clusters. Potential penalties are incurred only inside the source and receiving cluster for a sender-receiver pair from two different clusters.

We impose a restriction on receiving clusters with the use of a DBR. This results in data from a source cluster being limited to potentially one entry point to a receiving cluster at the DBR and one exit point at the source cluster at the DBR point of attachment. We show how this has the effect of increasing the path length in the source cluster and bound that penalty for the worst case.

We choose the simple solution for the receiver to join its closest MBR. We have stated that the potential for a path length penalty now exists. We also show how that penalty occurs and bound that penalty for the worst case.

1. Source Cluster Penalty

The source cluster penalty is a result of a single path used from a source to a receiving cluster for many senders and receivers. The penalty occurs when a DBR joins an MBR of a source cluster that is not on the shortest path from a sender to a receiver. This is illustrated in Figure 4.4. The difference between the path lengths for the receiving cluster MBR (δ) is small back to the source cluster. The DBR, C_n in this case, chooses A_n as its point of attachment. The sender is located at B_n, and the furthest B_n could be from A_n is the diameter of the source cluster, marked as *diam(C1)*. This is the worst case for a source cluster. All receivers in the receiving cluster pay a penalty of the diameter of the source cluster for this sender.

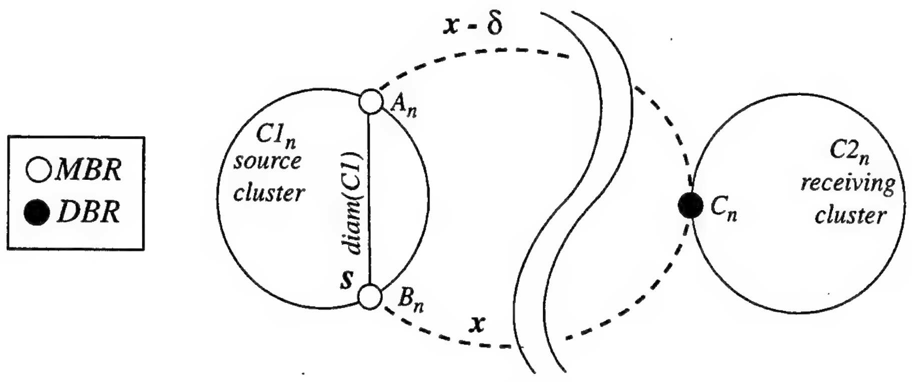


Figure 4.4: Source Cluster Penalty

2. Receiving Cluster Penalty

The receiving cluster penalty is a result of the receiver joining its closest MBR for a source cluster. The penalty occurs when the MBR is not on the shortest path back to the source cluster, which in turn happens when the closest MBR is not the DBR for the source cluster. This is illustrated in Figure 4.5. The worst case occurs when the receiver is approximately half the cluster diameter away from both the DBR and the MBR it is joined to and the MBR joined is a diameter away from the DBR. In Figure 4.5, MBR C_n is *diam*(C2) away from the DBR at B_n. The receiver joins C_n, as it is closer by ζ than B_n. Had the receiver chosen B_n it the data would have to traverse 1/2 *diam*(*C*2) to get to the receiver.

Instead, it traverses 1 and 1/2 times the diameter. Thus, the worst case penalty is a diameter.

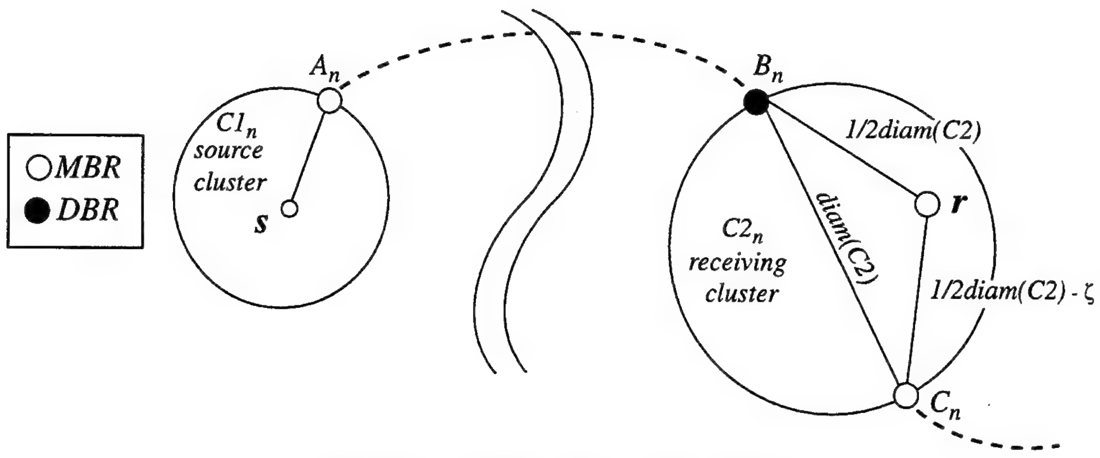


Figure 4.5: Receiving Cluster Penalty

3. Summary

The worst case penalty is composed of two parts for any sender and receiver pair. It is bounded by the sum of the source cluster and receiving cluster diameters.

The maximum penalty applies at any level, *k*, in the hierarchy.

$$MaxPenalty = diam(C_k(s)) + diam(C_k(r)) \qquad (4.1)$$

It is important to note that this penalty is applied to a specific sender and receiver pair, not to an entire cluster and not to an entire group. In Chapter VII, we will show that the maximum penalty is rarely reached. The more likely case is that some penalty is incurred but it is not near the maximum.

The value of the worst case penalty is dictated by the size of the clusters at their level in the hierarchy. The logical consequence is that smaller cluster sizes reduce the maximum penalty. However, the number of nodes in a cluster does not have a direct relationship with the diameter when the network has a regular topology. If a cluster is fully connected then the diameter is one hop. The upper bound on the diameter of an *n* node cluster occurs when the nodes are connected linearly. The bound is *n*-1 hops. One cannot state, with any accuracy, a relationship between the diameter and number of nodes for an irregularly connected cluster.

We next examine methods to reduce the worst case penalty. As expected, any improvements come at the cost of some additional infrastructure.

V. IMPROVING THE WORST CASE

We have shown the worst case penalty in the last chapter to be the sum of the source and receiving cluster diameters. The makes predicting the worst case difficult as it requires knowing the diameters of the clusters. While it might be practical to predict or measure the diameter of a leaf level cluster, the diameter of a cluster becomes more of an unknown as the scope of the multicast increases and the clusters are at the upper levels of a hierarchy.

The DBR election process is based on inter-cluster distances. All receivers in a receiving cluster pay the source cluster penalty for a sender. If there is only one or a few senders in a source cluster, the DBR for a receiving cluster could be attached at an MBR on the source cluster which is poor for all or most of these senders. This is due to the proximity of the senders to the MBR which is the DBR point of attachment on the source cluster. There may be another source cluster MBR with shorter paths to most of the senders internally with a distance to the DBR which also happens to be equal to or very near to the shortest path chosen by the DBR. Choosing this other MBR on the source cluster would reduce the source cluster penalty for all receivers in the receiving cluster.There is, at present, no way to know the location of this other MBR.

In this chapter, we propose methods to improve the worst case receiving cluster penalty and to reduce the source cluster penalty. We show how we can reduce the receiving cluster worst case to the diameter of a leaf level cluster. While we cannot reduce the worst case source cluster penalty for all senders, we introduce a method to reduce the average penalty, especially for those cases mentioned above.

In order to improve on the architecture presented thus far, we need to assume additional infrastructure or provide additional information. We look at each case individually and comment on the practicality of our proposed solution.

A. SOURCE CLUSTER SIDE

1. Additional Infrastructure

On the source cluster side, we do not attempt to reduce the worst case source cluster penalty. Under this architecture, with the receiving cluster making the decision about which MBR of the source cluster to attach to, the potential of realizing the worst case is always present. However, we can provide the receiving cluster with additional information about the senders inside of the source cluster to guide the DBR election process into making a more informed decision. This is particularly helpful in the case of multiple MBRs in both the receiving and source clusters which have identical or near identical path lengths. Presently, we use a simple tie breaking procedure for the receiving MBR in which the lower address becomes the DBR. This proposal injects intelligence into the tie-breaking process.

The idea is to provide a Measure of Goodness (MOG) for a source cluster MBR. All MBRs of a source cluster are aware of the senders within their own cluster. We propose that each MBR calculates the distance from itself to each of the senders in its cluster. This can be done for senders in each group or senders in all groups in that cluster. Calculation of the MOG in a per group fashion increases the MBR overhead for the DBR election in that several MOGs must now be computed for each source and sent to the other clusters on the control tree. Alternatively, including all senders for all groups generates less traffic on the control tree.

Each MBR averages over the number of senders to arrive at an average distance to each sender. Two types of MOG could be defined. For example, the Measure of Goodness for an MBR per group could be:

$$MOG_G = \frac{1}{|S_G|} \sum_{s \in S_G} d(MBR, s) \qquad (5.1)$$

where S_G is the set of senders in group G and $|S_G|$ is the number of senders in group G. Similarly, we define the MOG for all cluster senders to be:

$$MOG_C = \frac{1}{|S_C|} \sum_{s \in S_C} d(MBR, s) \qquad (5.2)$$

where S_C is the set of senders in the entire cluster, regardless of which group they belong to and $|S_C|$ is the number of senders in a cluster C.

Once each MBR has computed its MOG, the information is passed to the registrar for this level to be stored with the MBR list it already maintains. When a DBR election needs to take place and the registrar passes the source cluster MBR address information on the all-MBR tree, they also pass a MOG associated with each source cluster MBR.

2. Use of the MOG to Reduce the Penalty

To use the MOG, we modify the DBR election process. The modified algorithm is in Figure 5.1. The list of source cluster MBRs contains the MOG data. The distance to the source cluster MBRs and the MBR's MOGs are input into some function to compute an overall figure of merit (FOM) for a source cluster MBR. Since the lower number is better for both the distance to the source cluster and the MOG, the function could be as simple as an add. Then the FOM would be the average distance from a receiving cluster MBR to the senders in the source cluster.

```
     Modified DBR Election at a level n MBR
2  Receive list of source cluster MBRs and their MOGs on the all-MBR tree
3  Calculate distances to source cluster(s) MBRs
4  Compute FOM = f(MOG, distance)
5  Exchange FOM with other MBRs in my cluster on the All-MBR tree
6  If (my FOM is lowest) then
7      Join the source cluster MBR with lowest FOM
8      Notify all other MBRs in my cluster that I am the DBR
```

Figure 5.1: Algorithm for DBR Election using MOG

We illustrate how the MOG can reduce the penalty using Figure 5.2. We place three senders in the source cluster C1. The DBR for cluster C2 chooses MBR A to attach to since the inter-cluster distances are equal and A is lower than B. If this is allowed to happen, the receivers in cluster C2 incurs a source cluster penalty of 3 hops for both S1 and S2. No penalty is incurred for S3. Table 5.1 shows the MOG calculations for this case. MBR B is chosen if the MOG is used and the overall penalty to receivers in C2 is reduced. Only the sender at S3 incurs a source cluster penalty of 1 hop. This also illustrates that the use of the MOG does not eliminate the source cluster penalty and so cannot improve on the worst case for any one sender, but it can improve overall performance at the additional cost of calculating and storing the MOG.

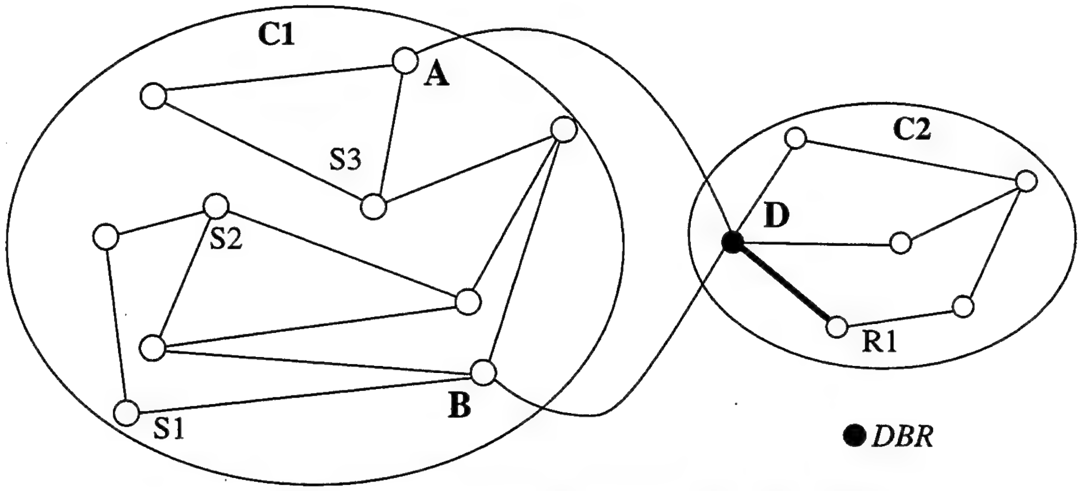


Figure 5.2: Sample Clusters to Illustrate use of the MOG

Table 5.1: MOG Calculation

MBR	S1	S2	S3	MOG
A	4	4	1	3
B	1	2	2	1.67

3. Practicality

This proposal is similar in concept to the schemes in Reference [21] for Closest Entry Routing (CER) and Overall Best Routing (OBR). These two aggregation schemes used in hierarchical unicast pertain to the amount of internal routing information that would be propagated outside of a destination cluster. With CER, no routing information describing the internal behavior is propagated outside the destination's cluster. For OBR, an average estimated distance to all nodes in the destination cluster is made available [21]. We turn this around and apply it to the sending side and refine it to include an average distance to the senders rather than all nodes in a cluster.

This proposal appears practical from an implementation viewpoint since each MBR needs to know where the senders are anyway in case it becomes a point of attachment for a DBR. However, knowing the address of a sender and knowing the distance to the sender are not the same. The distance to a sender may not be in the MBRs tables which means it would have to obtain it, probably from the leaf level MBR. In the case of a cluster high in the hierarchy where the MBR and the sender are not near each other, this could become an unreasonable burden for an MBR. This is especially true if the number of senders is large. It is more likely, therefore, that the group specific MOG, MOG_G, would be used over the cluster MOG, MOG_C.

Before we decide if we should use the MOG in the DBR election process, we need to examine if it is really necessary. Based on the simulation results (Chapter VII), it appears that the maximum source cluster penalty is not incurred too often, if at all. The case of a single sender in a cluster with many receivers far away, such as in a distance learning environment, appears to be the ideal case for this type of improvement. The option should be available, as we said before, on request.

B. RECEIVING CLUSTER SIDE

1. Additional Infrastructure

In this section, we propose a method to reduce the worst case receiving cluster penalty to the diameter of a leaf level cluster. In order to do this, we assume additional infrastructure in the form of a congruent unicast hierarchy. A congruent unicast hierarchy implies that the same hierarchical clustering and routing that we describe for multicast exists for unicast. This is not true in the present Internet. We also assume that the unicast hierarchy ensures that routing traverses the hierarchy strictly. All MBRs for every level in the hierarchy are co-located with the unicast border routers (UBRs) for the same clusters.

2. Use of the Unicast Hierarchy to Reduce the Penalty

In the unicast hierarchy, we assume that the border routers at different levels have a similar relationship to border routers in the present Internet. Similar to the way that they are used to provide reachability information about members in their domains, they provide reachability information about nodes in their clusters [43]. That is, the border routers can provide routing information about the nodes in their cluster. Given this hierarchical nature, the UBRs provide shortest path information between UBRs at different levels of the hierarchy.

The actions of a receiver join are modified to do the following:

- a receiver joins its closest leaf level MBR,
- a leaf level cluster DBR is elected,
- the path this DBR gets from the unicast hierarchy will contain a level *n* MBR due to the congruent unicast hierarchy.
- this level *n* MBR would be the DBR if no unicast hierarchy existed.

Thus, the shortest path in the unicast hierarchy will pass through the MBR that our present scheme would choose as the DBR (disregarding the MOG case). This amounts to reducing the worst case penalty for the receiving cluster to the diameter of a leaf level cluster, which we assume to be small.

We illustrate this in Figure 5.3. The receiver at *R* is deep in the hierarchy. Its closest level *n* MBR is at *B*. Without the congruent unicast hierarchy, *R* joins *B* and *B* joins the DBR at *A*. But with the modifications stated above, *R* would join its leaf level MBR at *D* and *D* would join the DBR at the leaf level. In this case, we assume *D* is the DBR. The shortest path to the source cluster that *D* receives from the unicast hierarchy goes through

A, which would have been the level *n* DBR for this cluster. The worst case penalty is reduced to the diameter of the leaf level cluster containing *R* and *D*.

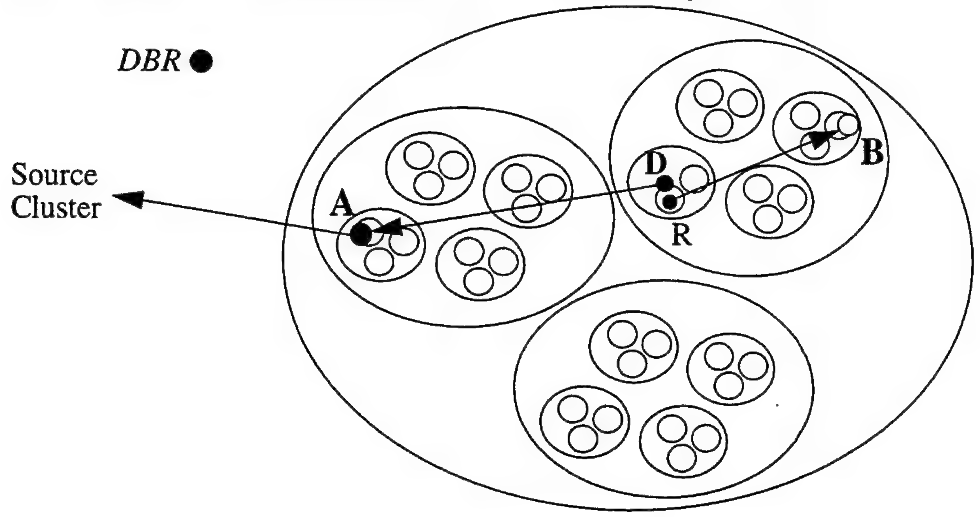


Figure 5.3: Reducing the Receiving Cluster Penalty

C. THE SINGLE SENDER CASE

We have shown two methods to reduce the penalty in the general case for a group with some arbitrary number of senders and receivers. There is one special case which should be addressed. This is the case of the single sender to a group with an arbitrary number of receivers. Based on the simulation data in Chapter VII, this special case warrants our attention. Since we have a single source cluster in this case, we have only one DBR per receiving cluster. It does not make sense to pay the receiving cluster penalty for receivers who join their closest MBR. If they are allowed to join the DBR for this source cluster, we eliminate the receiving cluster penalty. We cannot eliminate the source cluster penalty for reasons we have already elaborated on, but if we use the MOG as described above, we can approach the shortest path for almost every receiver in the group.

For this situation we propose a special notification message to the MBRs that this is a special group type. The MBRs would then notify the receivers to join the DBR instead. We then have only a minimal source cluster penalty, due to MOG, with a shortest path from the source cluster to all receivers in the group.

D. SUMMARY

We have examined the two components of the penalty and offered solutions to improve the performance in both cases. In one case we can improve the path length performance for a receiving cluster by reducing the sending cluster penalty. In the other case we can actually tighten the upper bound of the penalty by a significant amount. In the third case, we have a solution to a special group make-up which eliminates the receiving cluster penalty.

VI. NETWORK OVERHEAD

In this chapter, we discuss network overhead, a general term which refers to allocation and usage of the network components and their resources. Costs associated with network overhead are bandwidth consumption, router state information and traffic concentration [56].

We discuss the network overhead for CHARM primarily in terms of generated control traffic and router memory requirements resulting from state space required. The frequency of message transmission is discussed in terms of the expected frequency of related events and message sizes are given in terms of the number of items in each message. Most of the messages contain a group address (4 bytes for an IP address) as an item.

A. CONTROL TRAFFIC

We have introduced two types of control trees and discussed their functional responsibilities in Chapter III. We now examine these control structures with respect to the volume and frequency of the expected traffic. Level-specific information constitutes the primary traffic on the level *n* control tree while cluster-specific information is passed on the all-MBR tree. The reader may wish to refer to Chapter III, section E when reading this chapter as the functional details are not be repeated here.

1. Level *n* Control Tree

The level *n* control tree traffic is dependent on the frequency with which groups are created/destroyed and the frequency with which senders are added to a group. As described in Chapter III, it supports the replicated distributed directory service for group registration. The responsibilities of the registrar include passing level-specific group information to other registrars physically located in another cluster which has the same parent along the tree. The items of information passed are the group address when a group is first created and source cluster MBRs for the DBR election process. All registrars at a level have group address information for all groups in other clusters at their level with the same parent.

Group information is passed upon group initiation. Source cluster MBR lists are sent when a sender joins a cluster for the first time. Once a sender appears in a cluster, that cluster remains a source cluster as long as there is some sender active in it. It does not need to pass the MBR list again unless all senders in the cluster have been terminated.

In addition, the same cluster information (the MBR list) is used for the DBR election and it is passed between level n and leaf level registrars. This information is passed in response to a unicast query from a leaf level registrar whose return address is included. Thus, no traffic flows on the level n control tree when this occurs.

Since the frequency of this traffic is related to the frequency of group and sender creation, it depends on the collective group membership change activity at a given level.

The new source cluster notification is of the size of the source cluster MBR list (the order of the number of MBRs in a cluster times the size of their address) plus the group address and a source cluster ID. This message is passed once per cluster at this level unless network reconfiguration requires a new DBR election.

2. All-MBR Tree Traffic

The primary information passed on the all-MBR tree depends upon new sender joins regardless of whether they are local or external to a cluster. Either a new source cluster notification along the level n control tree or an internal new sender registration from the registrar cause traffic on this tree. In response to a new source cluster notification, a DBR election may take place.

Two phases occur in the DBR election which generate traffic on the all-MBR tree. First, the registrar injects the source cluster MBR list and second, each MBR passes its shortest distance back to the source cluster. The size of the first phase message depends on the number of source cluster MBRs. The second phase has as many messages as the number of MBRs of the receiving cluster. Each message is the size of a local cluster MBR address and its associated distance back to the source cluster.

A DBR election takes place once for each source cluster. Therefore, the frequency of this traffic is expected to be low.

Internal new source notification is passed on the all-MBR tree in response to a new local sender registering with one of the MBRs. This MBR multicasts the new sender identification and the related group address on the all-MBR tree. The MBRs then pass the information to any local receivers that may be attached for this group. The size of this message is two IP addresses and the frequency is dependent upon how often new senders come into existence after a group has been formed.

Periodic refresh of routing information and group membership may result in a change of DBRs (see section B). The all-MBR tree is used to synchronize the transfer of one DBR to another. The registrar is responsible for signalling the start and completion of the hand-off using the all-MBR tree.

The main advantage to having two control tree structures is that local cluster control information stays local to the cluster. For groups whose membership changes quickly, the impact is limited to the local cluster.

3. Interaction Between Leaf Level and Level *n* Registrars

The registrar at the leaf level registers groups and then passes the presence of this group to the appropriate level registrar. The leaf level registrar stores the name and address of the group and any senders' addresses who join. When a leaf level registrar gets a group creation request, it sends a unicast message to the registrar at the requested level of the hierarchy with a group address request and its own address as a return address. The level *n* registrar stores the group address, returns it to the leaf level and forwards it on the control tree.

Additionally, when a new receiver joins a group it queries the leaf level registrar for the level *n* MBR list. The leaf level registrar forwards the request to the level *n* registrar, again with its address included. The level *n* registrar returns the MBR list. The frequency of these messages is dependent on the frequency with which new receivers join. However, this information can be cached at the leaf level registrar so that this interaction will occur much less frequently. The level *n* MBR list can be cached since the MBRs are not expected to change with the exception of network failures.

4. Designated BR Election

The DBR election process is triggered by the appearance of new senders in a cluster where there were none previously (see Chapter III, section E.3). It occurs for each new receiving cluster, source cluster pair. The maximum number of elections that take place is bounded by $n(n-1)$, where n is the number of clusters at a level with the same parent. These do not occur all at once, rather they take place as new groups and senders are created. In addition, DBR choices are expected to remain static, once determined, as all members of all groups need to leave a cluster or a periodic DBR refresh must cause a change in the DBR occur before the corresponding state is explicitly pruned.

5. Registration

Two types of registration are required. First, new senders send to their closest leaf level MBR. Second, MBRs register with their level *n* registrar.

Senders are required to register for two reasons. First, they identify themselves for local receivers and MBRs to join. Second, the registration identifies the presence of a new source cluster, if the sender is the first one in the cluster for this group. They must be registered so that receiving clusters can join to them.

The MBR registration allows receivers to get a list of MBRs to choose from prior to joining the closest one. This is a static list that is reported once at MBR start-up.

B. PERIODIC REFRESH AND HAND-OFF

Network load changes dictate that routes change, new routes get added and old routes get dropped. We allow for a periodic refresh of distance information for DBRs which includes a periodic check on distances back to the source clusters. If the delay is substantially worse, then a new DBR election process is triggered and a hand-off to another MBR is caused by the present DBR's notification on the all-MBR tree.

As discussed in section A.2, the all-MBR tree is used to stabilize the hand-off procedure by synchronizing the events. If a hand-off is required, all MBRs attached to the old DBR join the new one before detaching.

C. STATE SPACE

State space refers to the amount of information that is required to be stored to support maintenance of trees in the network. There are three main functional components where state is stored: the registrars, the MBRs, and all routers in the network. The amount of state stored is dependent on the state maintained per level, the state maintained per group and the amount of state maintained per cluster.

1. Registrar

The registrar keeps level-specific and cluster-specific information. The purpose of the information stored is discussed in Chapter III, section E.2.

a. Level n

Sender information is maintained at the leaf level registrar and passed to the level n registrar so that it can be multicast on the all-MBR tree. No sender information is stored above the leaf level registrar and no receiver identities are stored anywhere. Group names are stored at the level n with the group address. A level n table entry is shown in Figure 6.1.

The amount of storage required for cluster-related information at each level n registrar ($n > 0$) is the size of a group entry, described in Figure 6.1 times the number of groups in that cluster. The source tag field indicates the presence of senders in a group in a registrar's cluster.

Group Address	Group Name	Source Tag

Figure 6.1: A Level n Registrar Group Entry

b. *Leaf Level*

A leaf level group entry is described in Figure 6.2. The upper bound on the amount of storage at the leaf level is the size of an entry times the number of groups.

Group Address	Group Name	Level

Figure 6.2: A Leaf Level Registrar Group Entry

2. Multicast Routers

The state required by multicast routers is the routing table information needed to properly route a multicast packet. A multicast router (MR) treats multicast routing internal to its cluster differently from external cluster routing since sources are aggregated outside of a cluster.

a. *Internal Sender State*

Source-specific trees, rooted at senders, forward multicast data to all MBRs for the cluster which are points of attachment for a DBR somewhere in the network. In order to aggregate the outgoing information from an internal sender, the MBR does not keep track of which senders use an external interface. When a DBR joins the MBR, group information is passed with the join. The MBR joins the senders as obtained from the local level *n* registrar for this group. The external link is now marked as an outgoing interface for this group. For every new join request that a DBR receives, it forwards a group information packet to the source cluster MBR so that it joins the appropriate senders.

The MBR which serves as a point of attachment receives multicast data internally which is stored as a (source, group) (*S*,*G*) pair in the internal routing tables. It encapsulates the packet and inserts its own address and cluster ID as the source information with the same group address before forwarding the data out on the external link. The outgoing link information is stored as a (source cluster, group) (*SC*,*G*) pair. The amount of state required to be forwarded on an external link of an MBR is *O*(*Gr*), where *Gr* is the total number of groups using this external link.

All routing for the source-specific trees within a source cluster is based on the internal router table state, which is (*S*,*G*) pair information. Thus, for group *G* inside the source cluster, the amount of state required is the sum of the number of senders in that group and for all groups in a source cluster, $\forall G \ \sum S_G$.

b. Internal Receiver State

The MRs are responsible for forwarding information to receivers for externally sourced multicast data. Data is forwarded from some source cluster point of attachment with (source cluster, group) pair (*SC*,*G*) forwarding information. The amount of state required to support this is *O*(*SCxGr*) where *SC* is the total number of source clusters at this level with the same parent cluster and *Gr* is the number of groups at this level. The MBR maintains a self (*SC*,*G*) entry, where *SC* is the source cluster the MBR belongs to, to make it possible for local groups to forward an internal source notification report on these links.

c. External Sender State

The special case of an MR with no internal group members is the external sender state. This is the case where the MR belongs to a transit cluster. The routing state information kept is no different than the internal receiver state. All routing state outside of a source cluster, regardless of the presence of members, is (*SC*,*G*) pair information.

d. MBR Cluster Related Information

In addition to the routing table entries, the level *n* MBRs maintain tables for source related information. A DBR maintains a group to source cluster mapping table and the other MBRs maintain a group to DBR table. An example of an entry in the group to source cluster table is shown in Figure 6.3. The amount of storage required for this table is bound by the number of groups at a level times the number of clusters at this level with the same parent cluster.

Group Address	Source Cluster ID$_1$	SC ID$_2$	...	SC ID$_i$

Figure 6.3: Level *n* Group - Source Cluster Mapping

An example of an entry of the group to DBR table is shown in Figure 6.4. The size of this table is bound by the number of groups at a level times the number of DBRs in a cluster. The group to DBR mapping is arranged by group since an MBR may have more than 1 DBR per group.

Source Cluster ID	DBR Address	DBR Address	...	DBR Address

Figure 6.4: Source Cluster to DBR Mapping

Two other tables are maintained by MBRs with source related information. They are called the "active" tables, the active *(S,G)* table and the active *(SC,G)* table. Their functions are similar. They keep information about active senders in their own cluster and active source clusters outside of their cluster which they are not presently routing. The active *(S.G)* table is used to forward local senders to receivers when they join their closest MBR. It is also used if an MBR should become a point of attachment for an external cluster so the MBR can join the senders. If the MBR does join, the *(S,G)* pairs are removed from the active table and used in the routing table. The only MBRs required to keep active *(SC,G)* pairs are the DBRs for a source cluster which has no receivers in the group. All other MBRs keep the DBR, group information mentioned above. When a receiver joins its closest MBR, the MBR joins the DBR and the entry is moved from the active table to the routing table.

D. TRAFFIC CONCENTRATION

Traffic concentration can occur when either multiple groups or multiple senders in a group multicast over a few common links. Studies done so far for center-specific trees over large groups, show by simulation that certain links may become bottlenecks and in the case of a large number of concurrent senders, traffic concentration may occur [56].

While there are no center-specific trees for data distribution in CHARM, there are shared inter-cluster links. These links are susceptible to traffic concentration. We do not force all cluster traffic to a single entry point in the receiving cluster since we allow for grafting to an inter-cluster tree for the same source cluster and group. However, we do not provide for multiple DBRs for a source cluster if the DBR links become congested.

1. Load Balancing the Inter-Cluster Tree

We propose a solution to the case where a single MBR becomes the DBR for too many source clusters. The load on the links when the MBRs measure their distances back to the source cluster may change drastically. This would be especially true if the same MBR is elected as the DBR for many source clusters. It is possible to add a load balancing factor into the DBR election process.

If two MBRs have the same distance to a source cluster, then instead of picking the MBR with the lowest address to break the tie, the one with the lowest DBR count is chosen. This could also be factored into the cases where the DBR count becomes very high and the distance measurement is close. In these cases, the DBR count becomes a factor applied to the result to make the election more evenly distributed among the MBRs.

VII. PATH LENGTH RESULTS USING RANDOM GRAPHS

Simulation of the proposed cluster-based hierarchy is important to verify the upper bound for the worst case path length penalty that we have established for CHARM and to examine how often this bound is reached, if at all. Towards this end, we have computed path length results via simulation for various multi-party interactions using random graphs.

Delay, measured in this simulation as hop count, is a major performance metric in the evaluation of our multicast architecture. Thus, we compare the paths provided by CHARM with the paths provided by SSTs for the random topologies generated. Such a comparison permits us to determine how frequent and severe the path length penalty is.

A. METRICS

The primary metric used to characterize CHARM is path length in hops. The choice of this metric is based on the following reasons. Firstly, we are interested in the quality of trees in a static scenario reflecting stable network conditions. Secondly, in the case of a lightly loaded network, hop count dominates delay across a path since the transmission time becomes negligible and factors such as router processing time and queuing delay, which directly depend upon the number of hops, dominate. Therefore, we measure hop count which is easy to compute and understand.

We note that, for the random topologies used, there is no correspondence between the physical distance between two nodes and the number of hops between them. Thus, a hop from Monterey to San Francisco is treated equivalent to a hop from San Francisco to New York City. In present-day networks, such situations are entirely feasible as long-distance leased lines and direct connections between geographically far-away nodes are common.

B. TOPOLOGY GENERATION

We simulate several random topologies and variations of groups which map to real-world applications that deploy multicast today. The random topologies were generated using a graph generation tool which permits generation of a two-level topology consisting of domains at the lower level and interconnections between domains at the higher level. The topology is specified by the number of nodes in a domain, the number of domains and the node degree of the intra- and inter-domain nodes. The output of the generation tool is a list of domain adjacency matrices and inter-domain links.

The simulator is constructed using object-oriented techniques to reflect a model of the Internet. The model consists of objects which represent network entities such as routers, routing tables, domains and border routers. In addition, there are several tools which operate on this model to generate least cost paths and trace the route between two nodes [4].

The model has been deployed for test and evaluation of tree construction techniques[51]. The code for the model is written in C++, as are all of the tools which construct trees on the model and the simulator runs under SunOS 4.1.3.

1. Graph Generation and Clustering

Graphs generated for testing purposes are composed of domains or leaf level clusters. All leaf level clusters are individually, randomly generated with a node degree of 4. The leaf level clusters are connected by inter-cluster links with two distinct node degrees between the clusters, one of 4 and another between 8 and 9. The two topology types, differing significantly in the inter-domain node degree, permit us to observe the behavior of CHARM in richly interconnected inter-domain topologies and validate our conclusions.

Three graph sizes are constructed from a single 192 node topology. The 192 node topology is trimmed down from 24 - 8 node clusters to 16 - 8 node clusters for the 128 node case and 12 - 8 node clusters for the 96 node case. In some cases, links have been added manually to insure connectivity of the clusters. The purpose behind simulating the different sizes obtained as a subset of a larger topology is primarily to show the effect of the depth of the hierarchy on the multicast. The scope of the 96 node case, for example, is equivalent to a lower-level scoped group operating in a topology of either the larger 128 node or the 192 node topologies. It is for this reason that, as far as the path-length performance goes, it *is* possible to simply focus on the top-level of a hierarchy and investigate only the groups operating at the highest level.

The leaf level clusters are kept at 8 nodes in all the topologies. Note that the node degree between 8 and 9 is applied only between the clusters. The network sizes in terms of the number of nodes are carefully chosen to permit as many clustering combinations as possible. Each cluster combination represents a hierarchy with the corresponding number of clusters at the top level.

Clustering above the leaf level is done manually by drawing boundaries around the 8-node leaf level clusters. The clustering scheme for each topology is supplied to the simulator by supplying cluster id's and parent-child relationships through an input file.

The diameter of the clusters is, in view of the worst case path length penalty bound, an important factor to consider for all the topologies. We note with emphasis the effect of clustering on the random graphs we have used before describing the results. Dividing a cluster into sub-clusters does not necessarily decrease the diameter. A decrease in the diameter when a cluster is divided into sub-clusters appears to be a logical consequence assumed in Reference [32]. For the topologies we have used, the effect of removal of links to isolate a cluster has exactly the opposite effect on the diameter - the clusters can have a larger diameter than the diameter of the graph and therefore the maximum path length penalty may actually go up. We note here that, to our knowledge, there is no study reported in the published literature that established a relationship between the diameter of a graph and its number of nodes for random graphs.

For the random graphs simulated, the sum of the two largest cluster diameters is plotted in Figure 7.1.

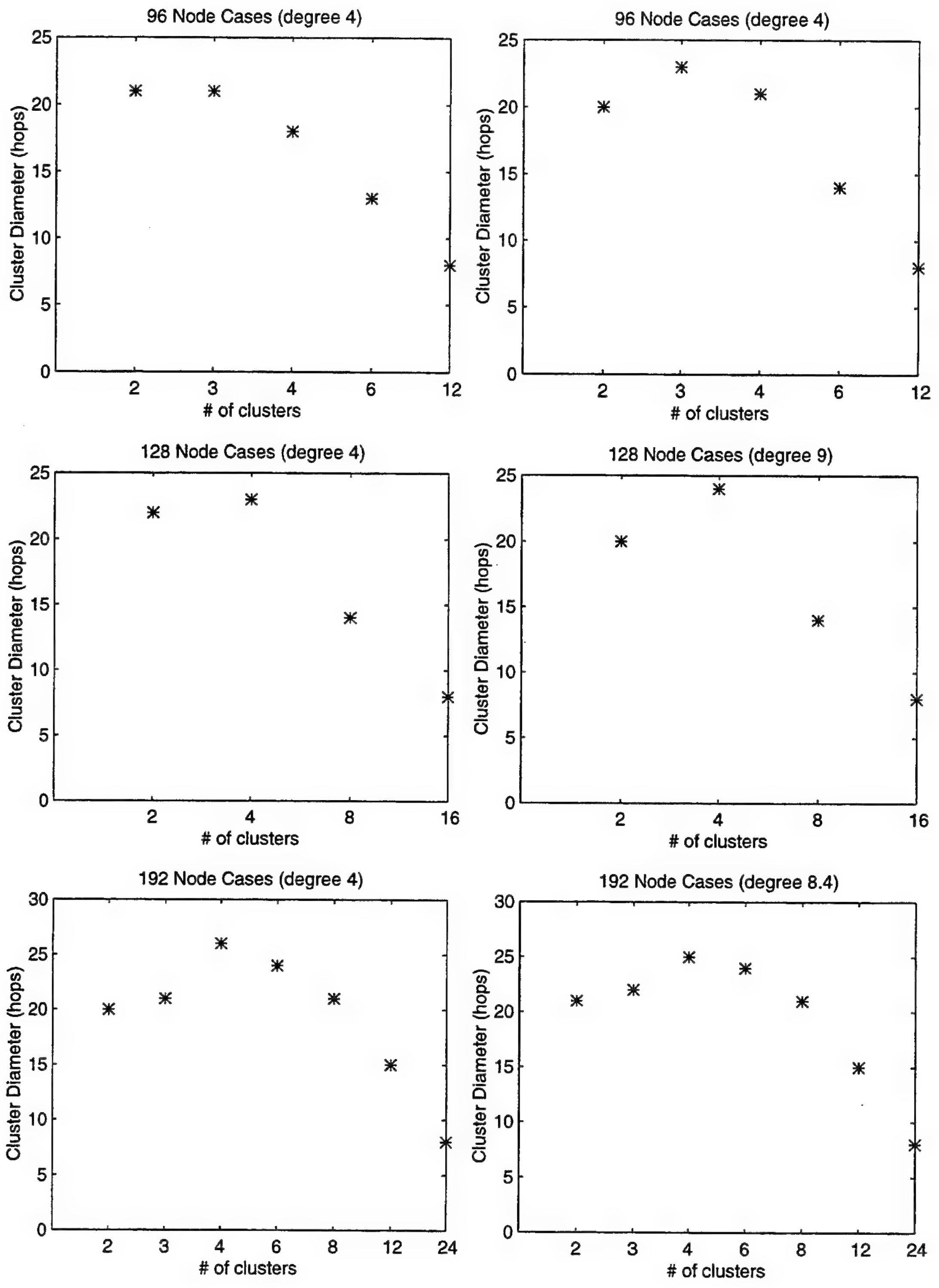


Figure 7.1 Sum of the Two Largest Diameters

C. SIMULATION SET-UP

The simulation input is a group of members, their location, and their role in the group as a sender or receiver. The clustering used adhered to the following rules established in Chapter III. All clusters of a level are connected. The shortest path of a cluster lies within that cluster. This is manually enforced in the simulation by artificially inflating inter-cluster link costs while calculating the shortest path between two nodes of a cluster. The effect that this has on the simulation is that occasionally when a source-specific tree is built by the hierarchy, it may not reflect the actual shortest path, which may be via another cluster. The result is that there are cases when a penalty is paid even by receivers internal to the source cluster which would not normally be.

The simulation handles static analysis of path lengths. We do not simulate group dynamics by way of membership changes. We also do not account for changes in the path lengths making the trees, once constructed, static.

Although queueing delays play a significant part in routing, they are not considered here. The queueing delay is typically related to link loads and processing capacity at intermediate routers. It is difficult to simulate the relationship between link load, processing capability and queueing delay accurately in a simulation without making it unmanageably computation-intensive for reasonably large topologies.

We do not simulate traffic concentration as discussed in Chapter VI, section D as it requires simulation of multiple groups, which we do not incorporate.

D. EXPERIMENTS

1. The Electronic Classroom

The electronic classroom is a special case of a single sender to a group of receivers spread out across a wide geographic area. It is also known as distance learning, where one location multicasts to a group of receive-only participants. This scenario is simulated to be able to isolate a single sender and see the effects of the hierarchy on a randomly selected individual sender to a random, moderately sized group of receivers.

The sender selected is the same for all cases reported. There are only 2 or less receivers per leaf level cluster. The numbers of receivers for three different size topologies are 24 receivers for the 96 node network, 32 receivers for the 128 node network, and 48 receivers for the 192 node network.

The results from this scenario reinforce the need for the improvement proposed in Chapter V, section C for the special case of a single sender. The reduction of cluster diameters reduces the path lengths for most receivers to nearly the shortest path. However, there are some receivers that incur receiving cluster penalties that are not necessary if we use the proposed improvement.

In Figure 7.2 through Figure 7.7, we plot the mean, median, and the standard deviation of the path lengths obtained from CHARM normalized with respect to the path lengths obtained from the source-specific tree as the number of clusters is changed. The general pattern in the path length plot (plot (a) in each figure) is that the initial clustering, the 2 cluster case, does well for path length performance. This is primarily due to the fact that half of the receivers get their data along the shortest path. As the number of clusters increases, the path length penalty also increases initially. As indicated by Figure 7.1, the cluster diameters increase at first before going down. This phenomenon is attributed to the loss of links in the process of clustering and is the reason why clustering performs worse initially. Although the actual value of the penalty goes down as the number of clusters increases sufficiently, the number of receivers that incur a penalty goes up. This is due to the fact that, with the increase in the number of clusters, there is a greater likelihood that a receiver is forced to a path along the hierarchy instead of finding the shortest one to a sender within the cluster itself.

The normalized penalty plot (plot (b) in each figure) shows the worst sender and receiver path length out of all senders and receivers in the group divided by the sum of the two largest cluster diameters, which is the worst case penalty. We note that the worst case penalty given is never exceeded in any of the cases.

Plots (c)-(f) in Figures 7.2 though 7.7 show how the penalty incurred varies between the different senders and receivers. We note that, as the number if clusters increases, the size of the penalty goes down, but the number of receivers that incur some penalty increases.

The lower node degree between domains leads to a better performance by CHARM. This is due to the fact that there are fewer choices for MBRs and fewer paths between domains. The result is an increase in the chance that the hierarchy will take the same path as the shortest path tree.

The electronic classroom scenario, being a single sender scenario, cannot be used to extrapolate the behavior of CHARM across multiple senders and the likelihood of pathological cases is higher. We note however, that, in every topology, for the largest number of clusters reported the hierarchy imposed by CHARM provides paths that are comparable to the shortest path for all receivers.

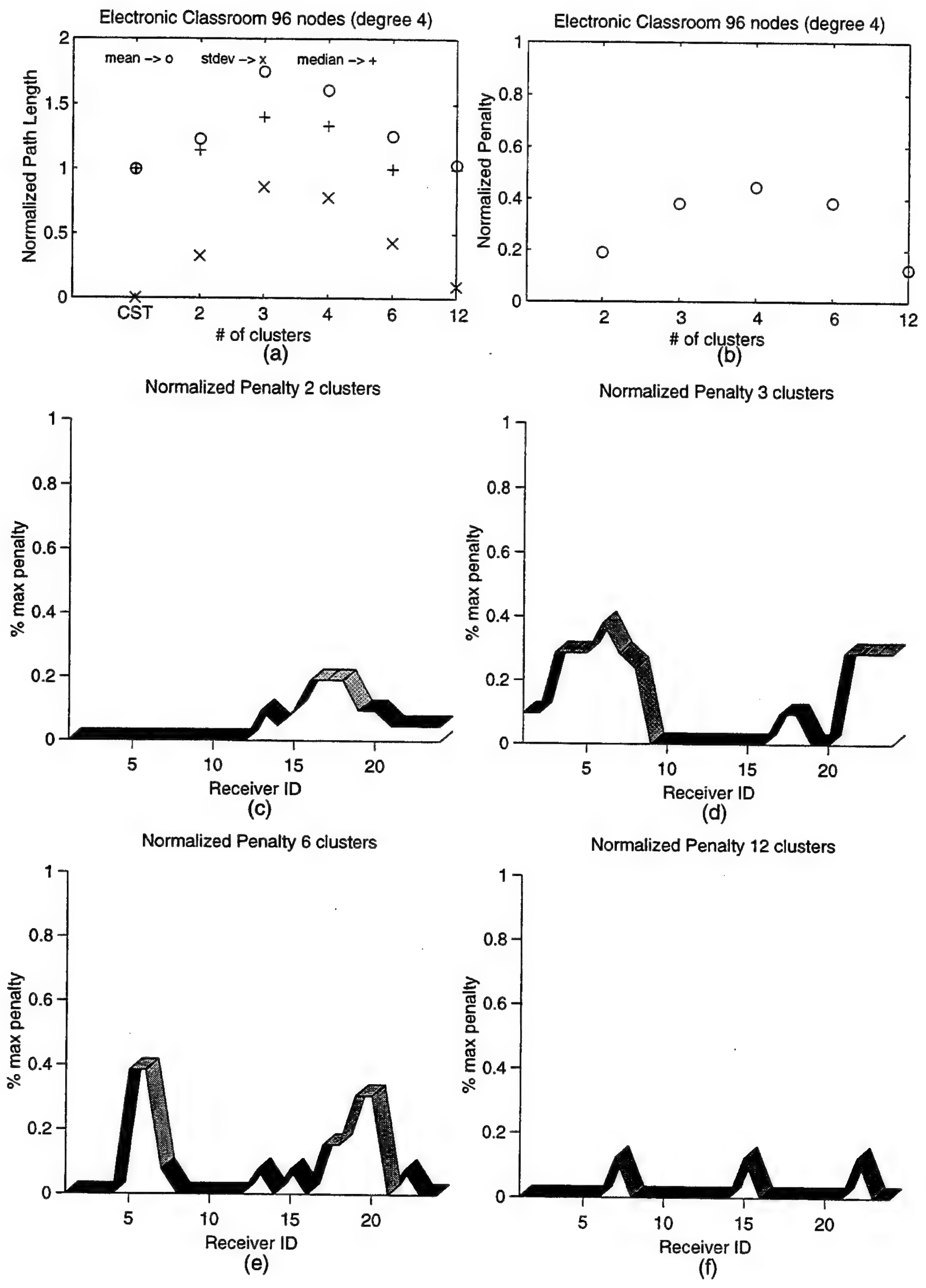


Figure 7.2 Electronic Classroom 96 Node Case Node Degree 4

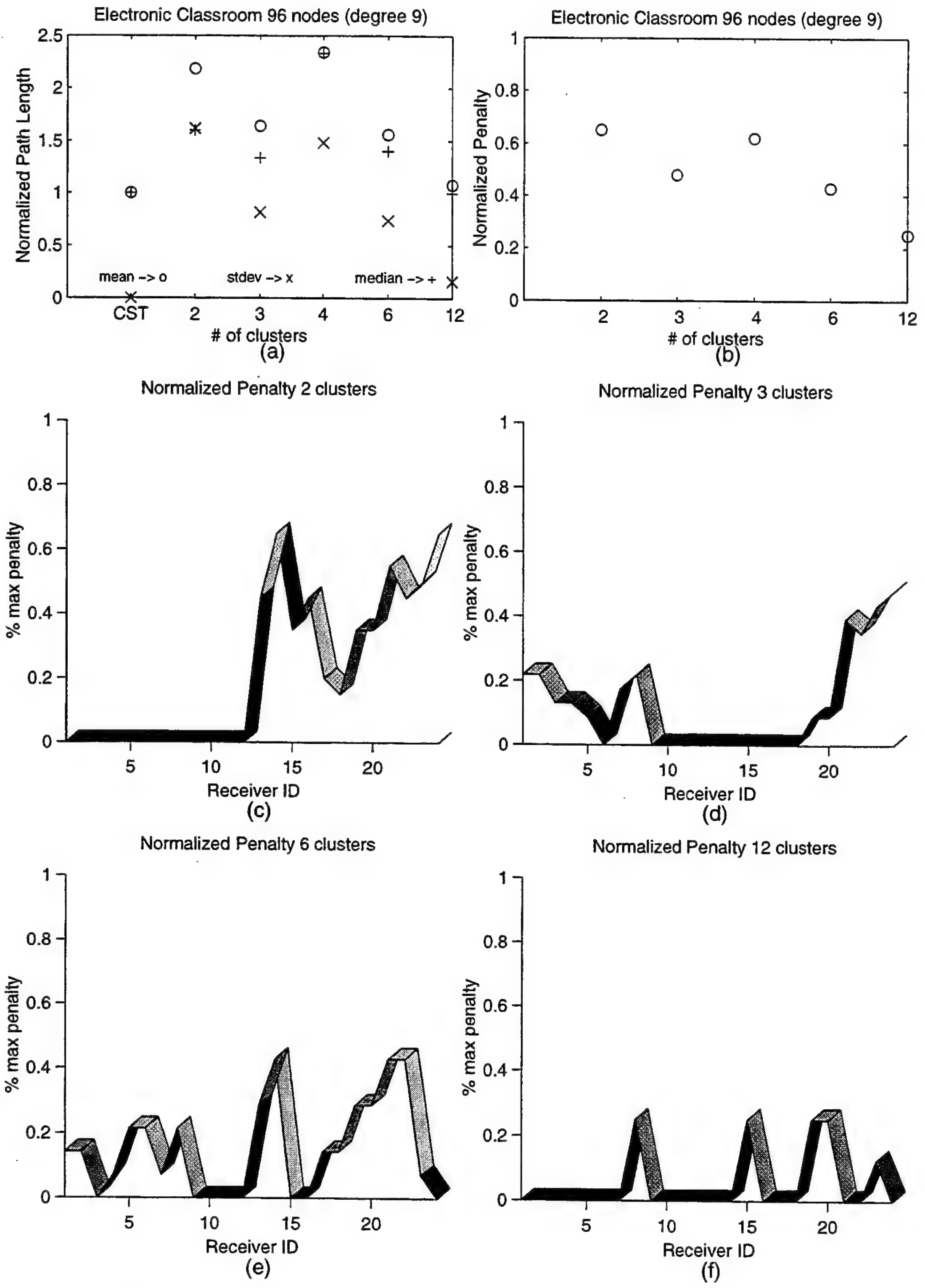


Figure 7.3 Electronic Classroom 96 Node Case Node Degree 9

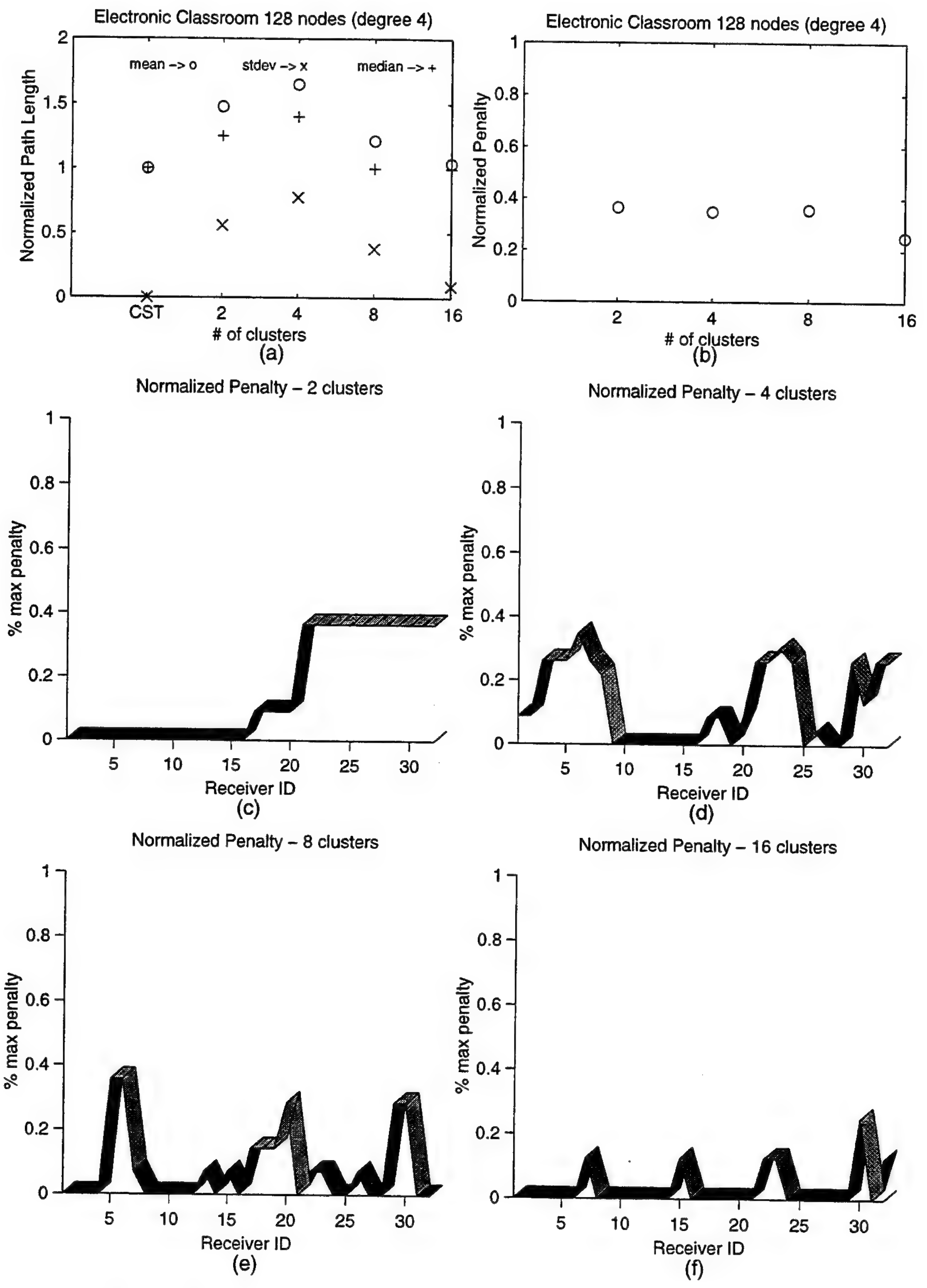


Figure 7.4 Electronic Classroom 128 Node Case Node Degree 4

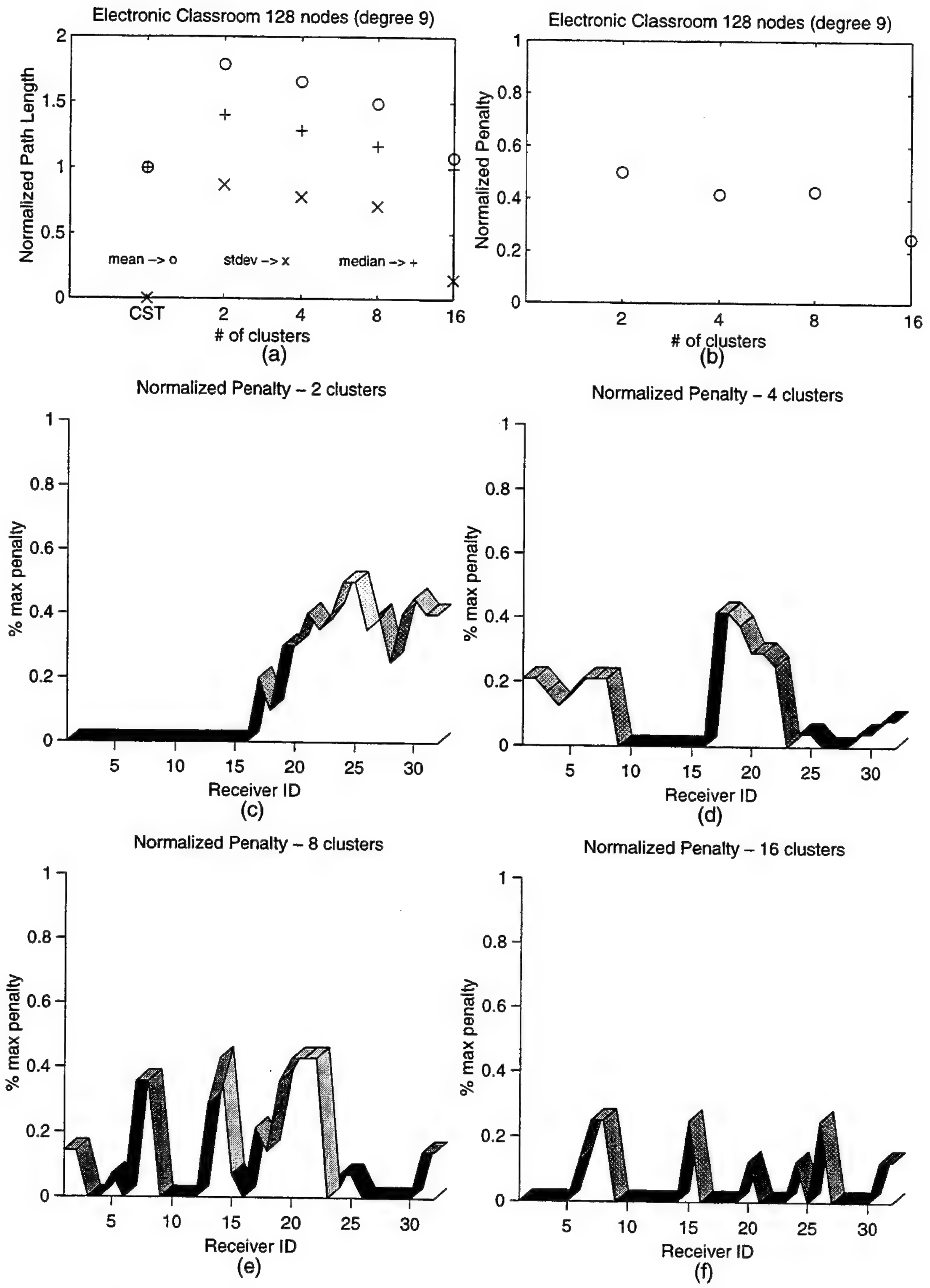


Figure 7.5 Electronic Classroom 128 Node Case Node Degree 9

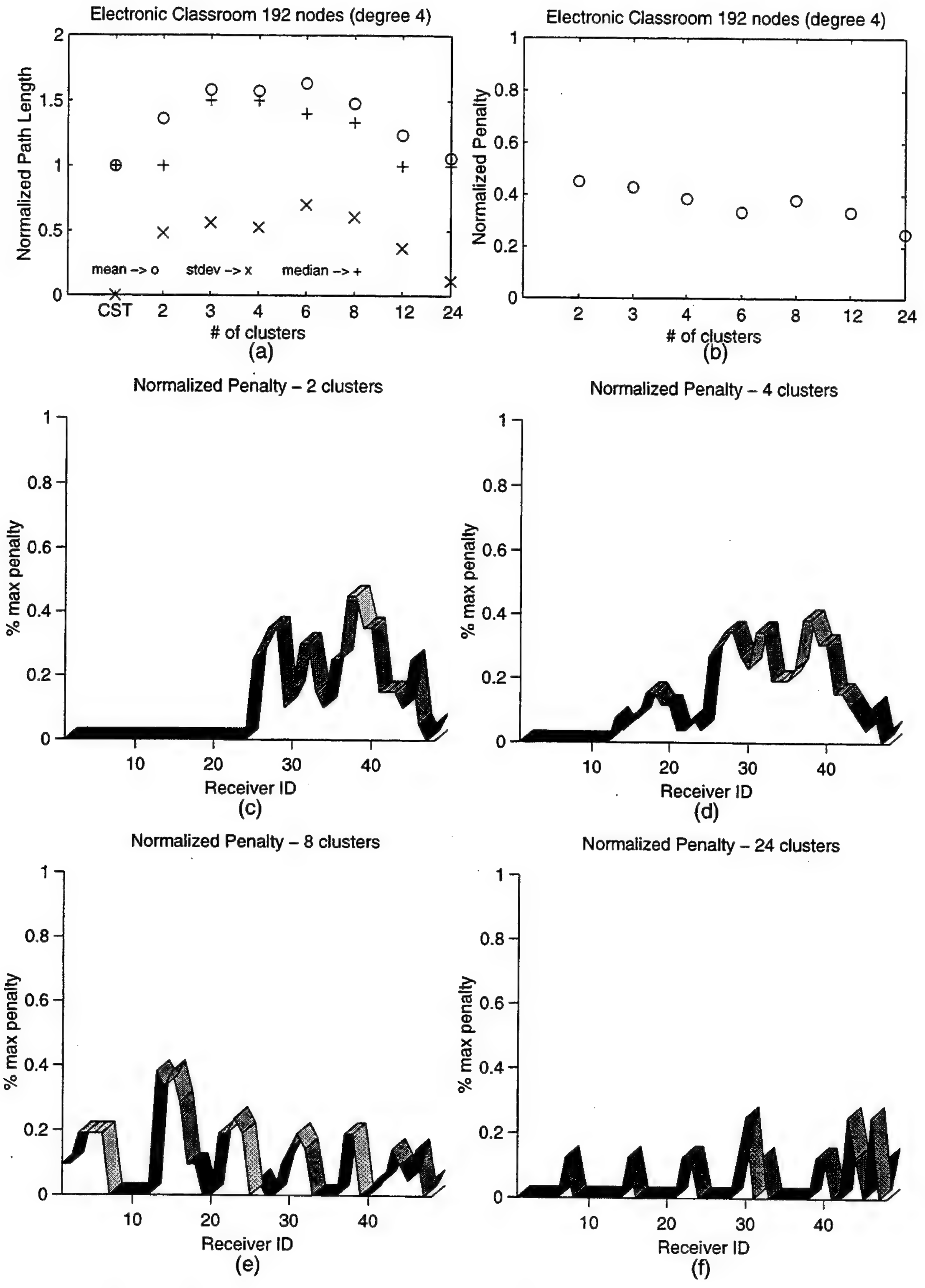


Figure 7.6 Electronic Classroom 192 Node Case Node Degree 4

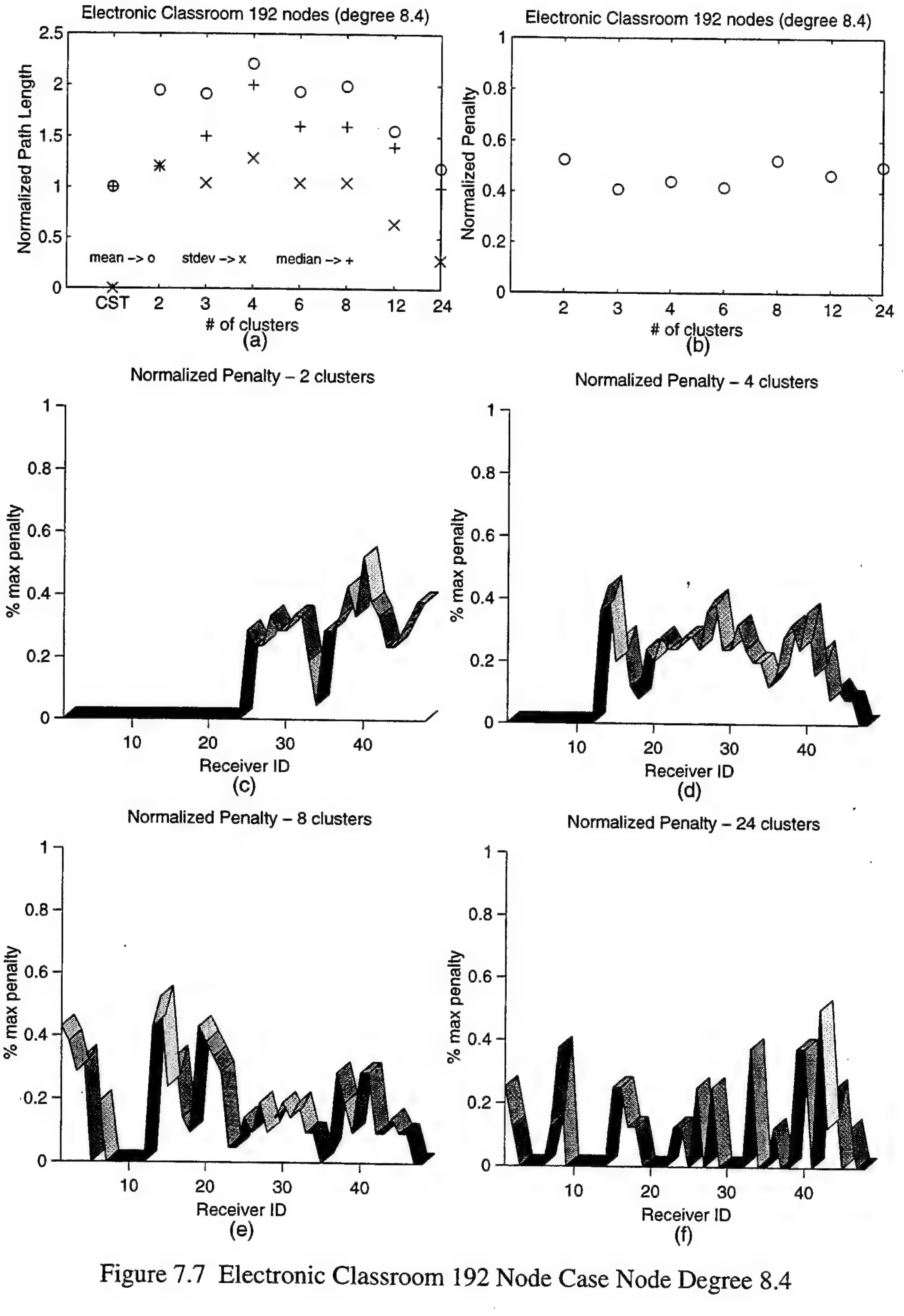


Figure 7.7 Electronic Classroom 192 Node Case Node Degree 8.4

2. Video Conference

The video conference scenario is a presently considered to be one of the primary applications of multicast. It is the case of multiple members in a group which are both senders and receivers with some receive-only participants. This scenario illustrates the effect of the hierarchy on a relatively sparse group where all senders are also receivers.

In the experiments reported, the number of receivers is fixed at twenty-five percent of the total number of nodes. The numbers of senders for three different size topologies are 16 senders for the 96 node network, 24 receivers for the 128 node network, and 36 receivers for the 192 node network.

In every case, the normalized path length approaches the shortest path as the number of clusters increases (plot (a)) and no pair of senders and receivers pays the maximum penalty (plot (b)).

The results also show that for the cases of node degree 4, the hierarchy achieves better path lengths than it does at the higher node degree. This is because with a lower node degree, the paths along the hierarchy are much more likely to be the same as the paths along the source specific trees.

Penalty patterns that are relevant to the effect of the hierarchy are displayed in each of the figures Figure 7.8 through Figure 7.13 in plots (c) through (f). The source cluster receivers which incur no penalty are seen as a patchwork of solid blue. In some cases where we force a sender to receiver path inside of a cluster, a small penalty is paid which disrupts the pattern and is indicated by an occasional light blue patch.

Other noticeable patterns are found in Figure 7.3 (c) and (d). Receiver 14 in the 2 cluster case and receiver 24 in the 3 cluster case both appear to have a poor choice for their closest MBR. This is evident in the striated colors indicating penalties. Similar patterns are found in other cases for senders. These patterns illustrate clearly that the source cluster and the receiving cluster penalties are sender- and receiver-specific. This is expected as the penalties are incurred only inside the clusters and are independent of how the source and receiver clusters are situated with respect to each other in CHARM.

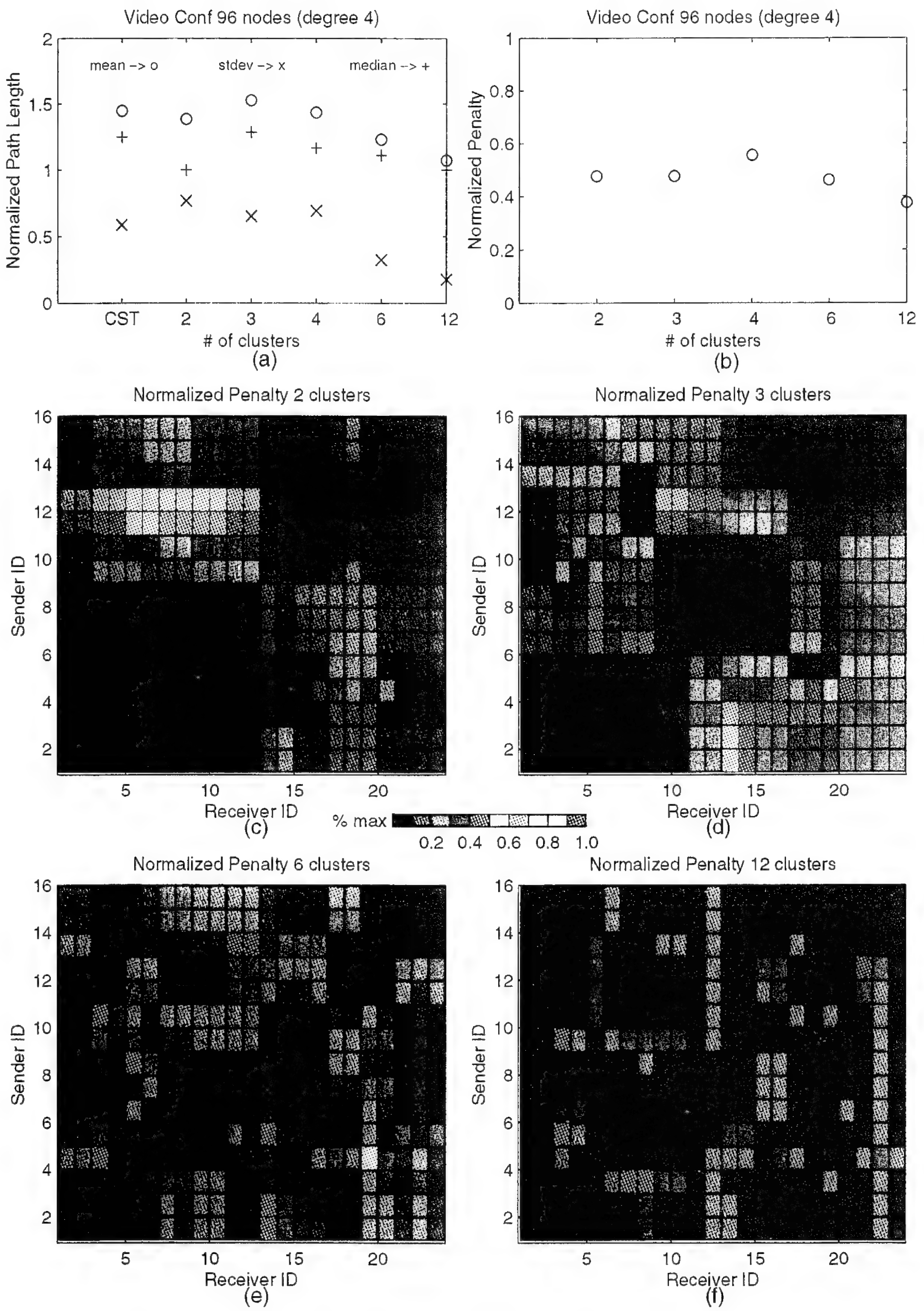


Figure 7.8 Video Conferencing 96 Node Case - Node Degree 4

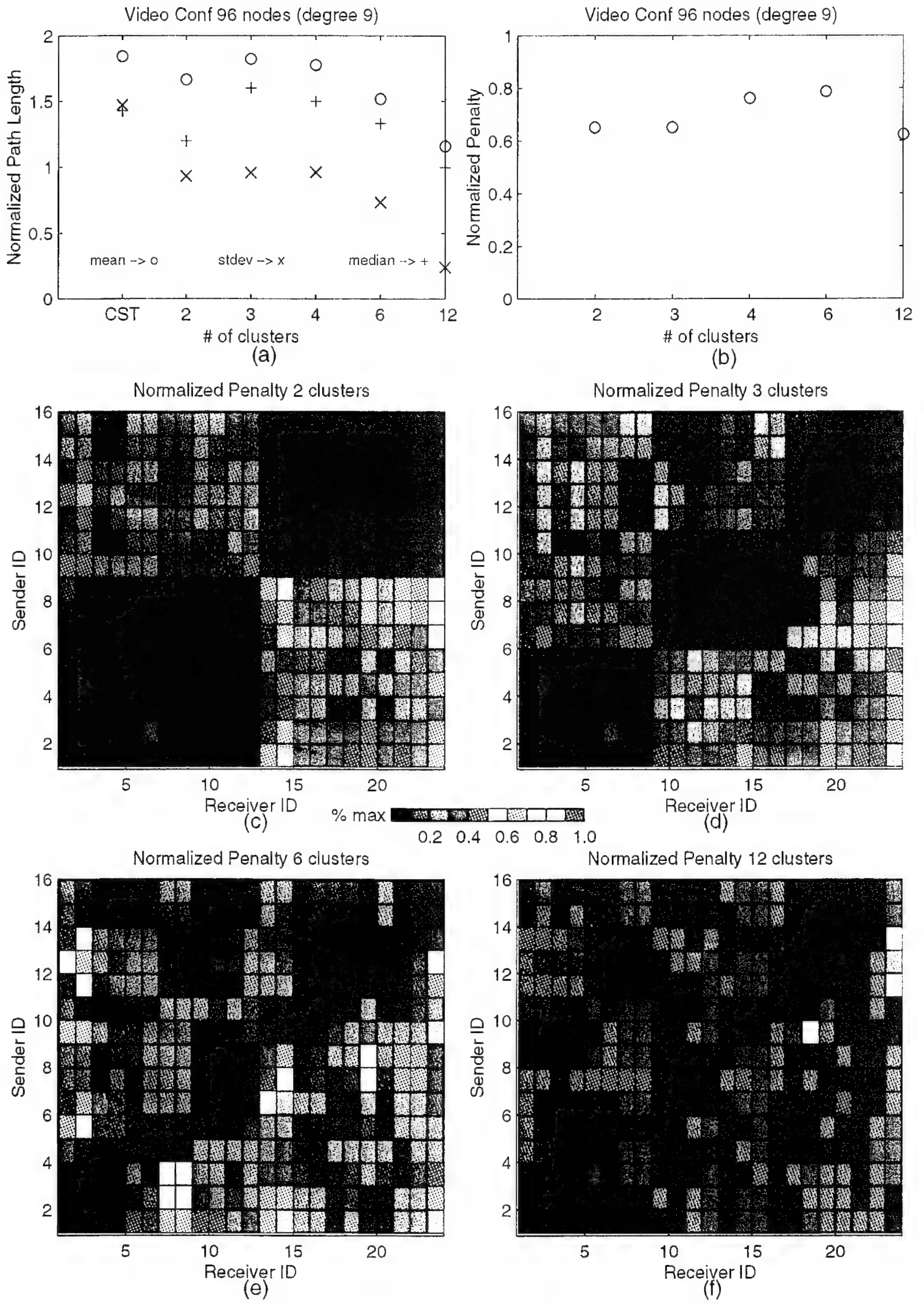


Figure 7.9 Video Conferencing 96 Node Case - Node Degree 9

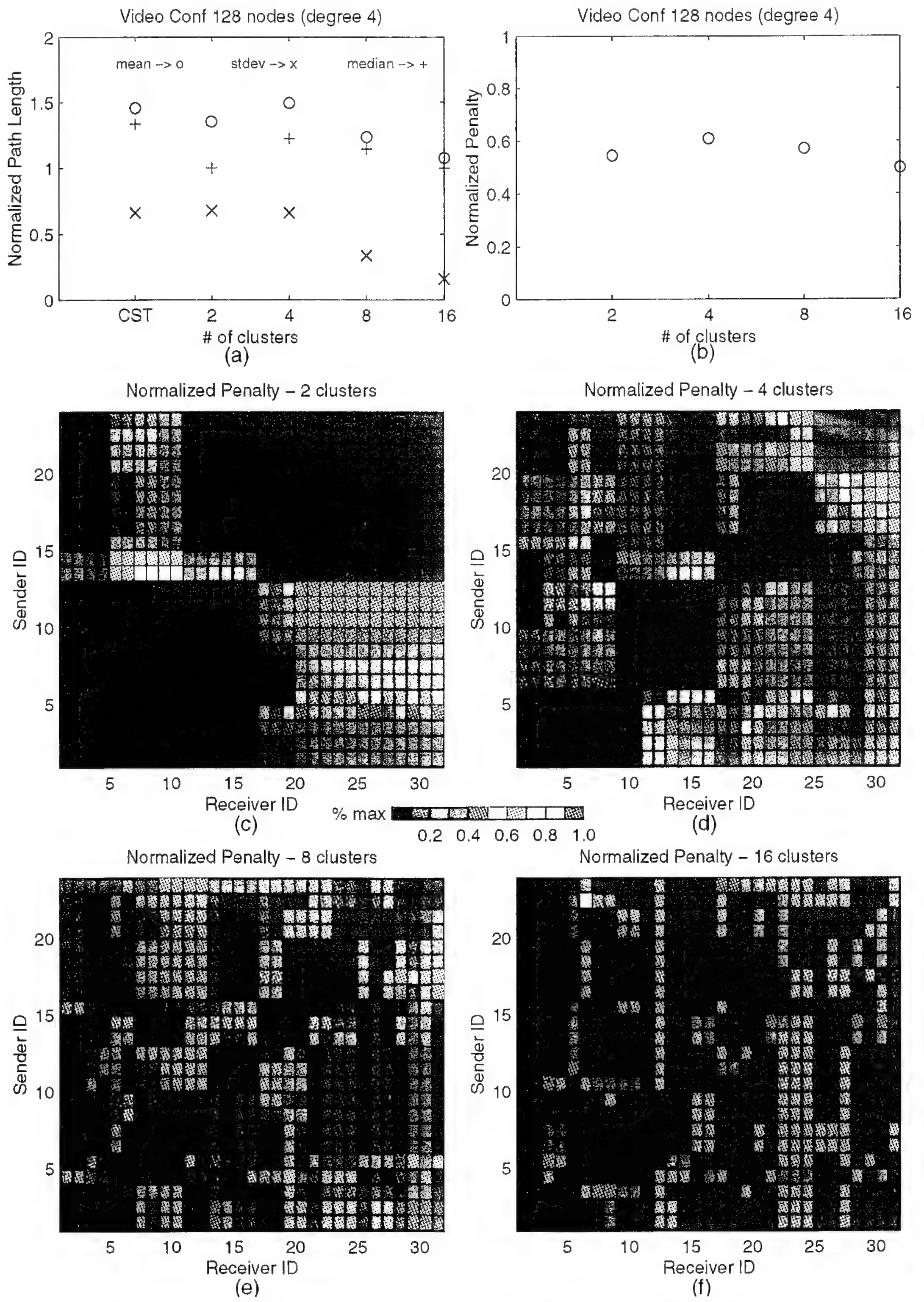


Figure 7.10 Video Conferencing 128 Node Case - Node Degree 4

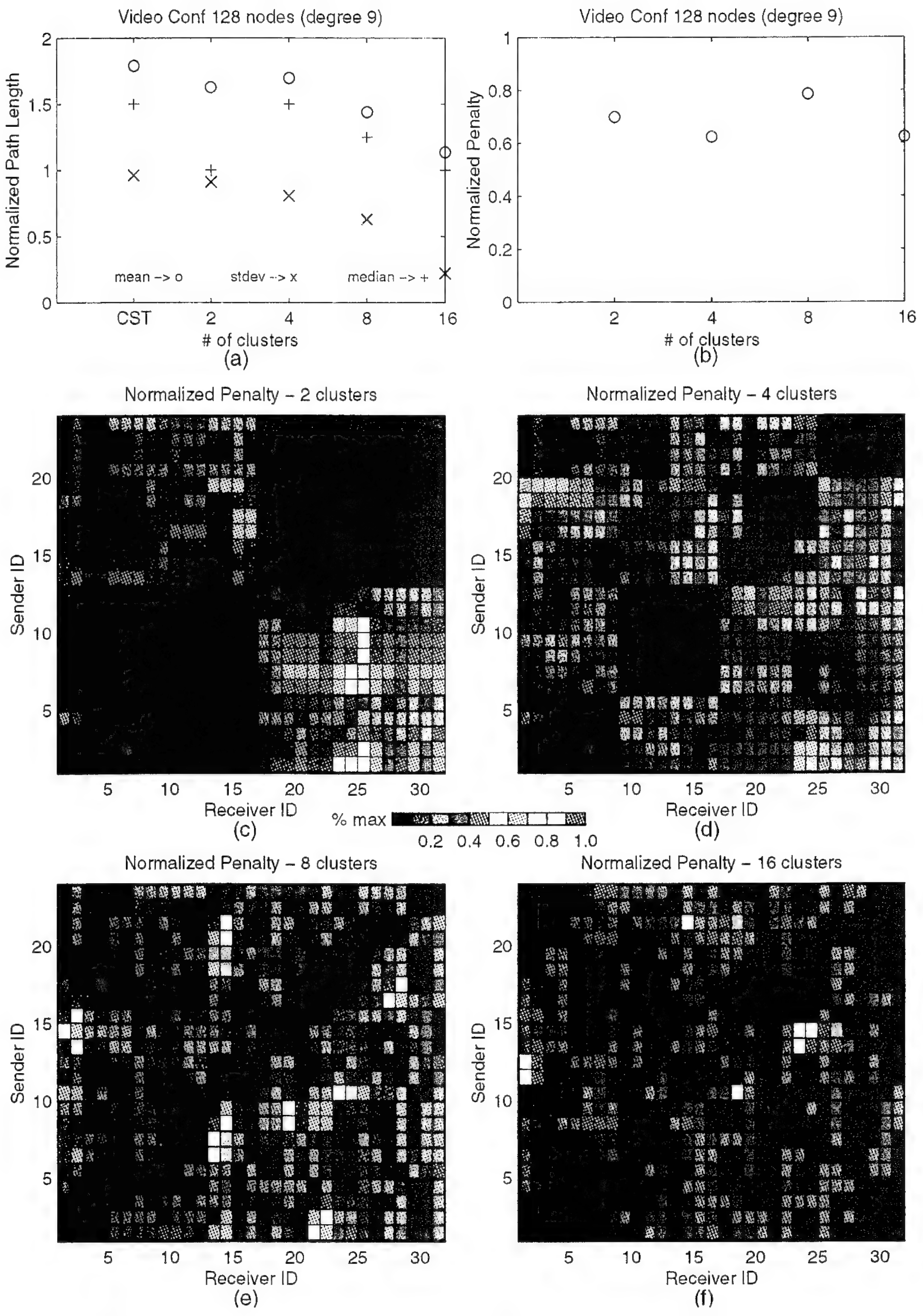


Figure 7.11 Video Conferencing 128 Node Case - Node Degree 9

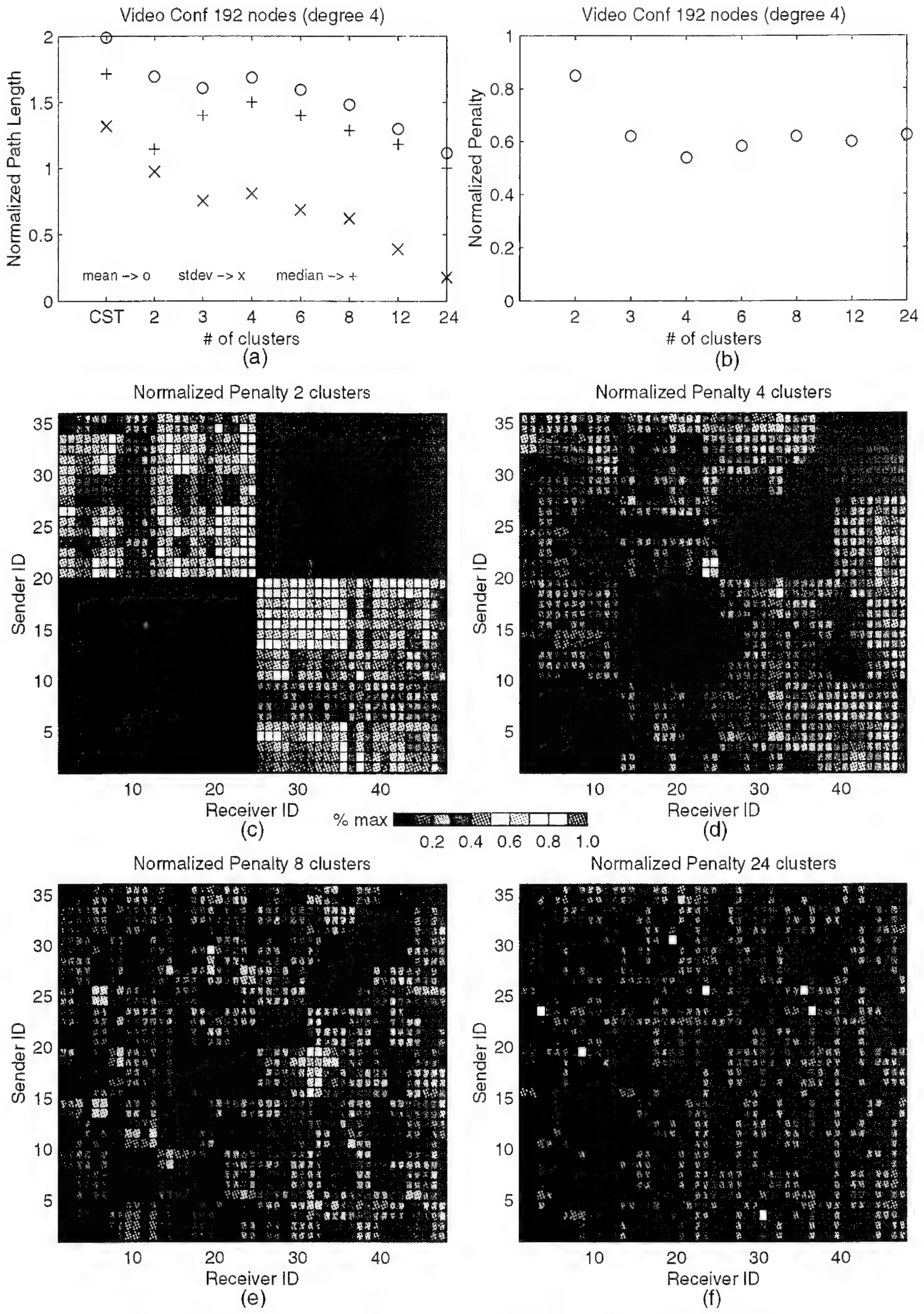


Figure 7.12 Video Conferencing 192 Node Case Node Degree 4

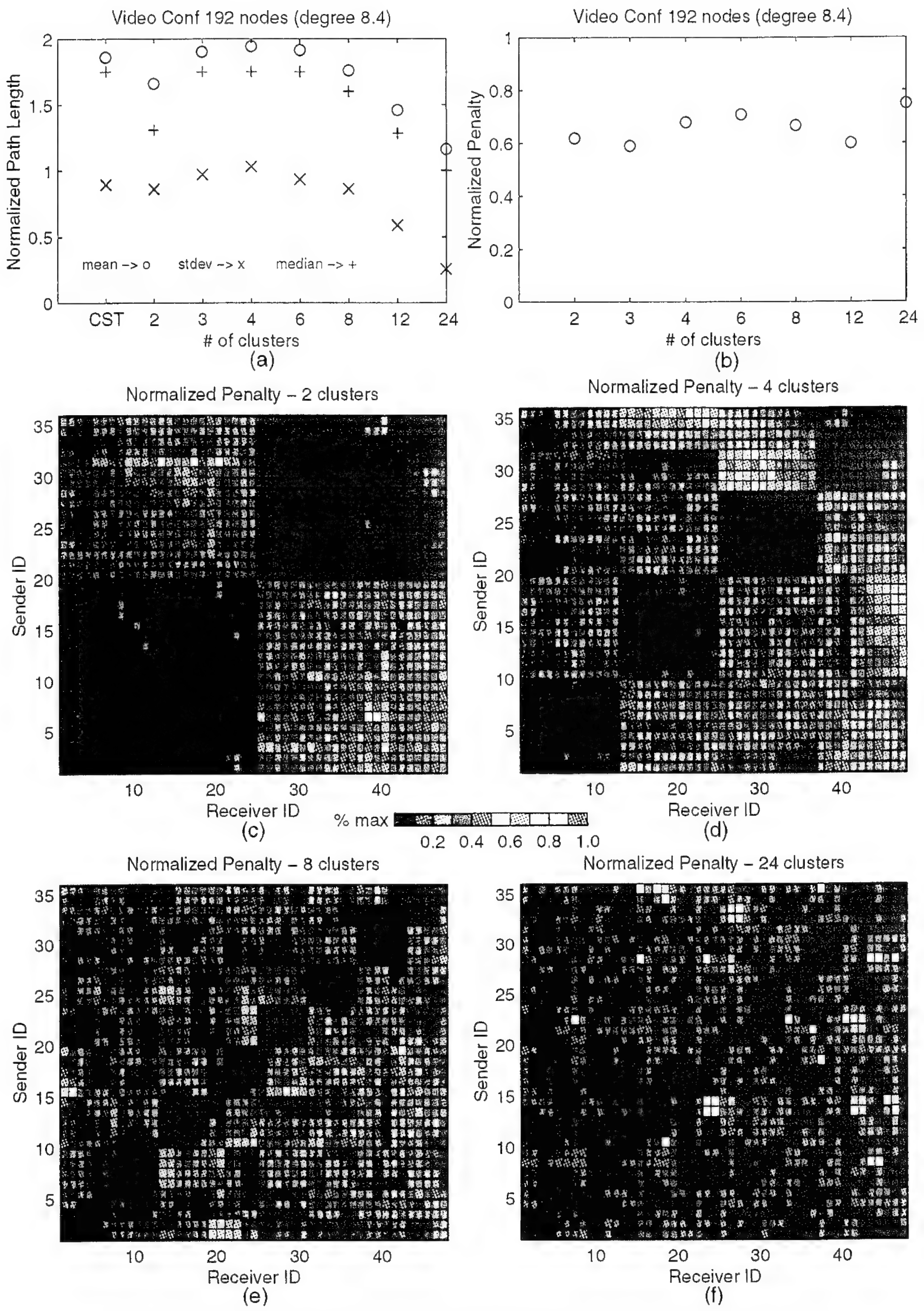


Figure 7.13 Video Conferencing 192 Node Case Node Degree 8.4

3. Distributed Interactive Simulation

The most demanding application for multicast is the case of a large group spread out over the entire network. This scenario is simulated to show the effect of the hierarchy for densely populated groups created by the distributed interactive simulation (DIS) community. DIS represents an application that requires widely distributed groups, a large number of members, and a large number of concurrent senders [36].

In this case, as well as the others, the smaller topologies are obtained by pruning down from the 192 node case to simulate the effect of operating at a lower level in the hierarchy. We observe in the DIS case, shown in Figures 7.14 through 7.19, that CHARM performs consistently well for all sizes of graphs in even a densely populated group scenario.

In the DIS case, group members make up 50% of the network. The purpose of such a high percentage for this scenario is to gain the widest cross section of the network and expose some pathological cases which return a maximum penalty incurred by some sender/receiver pair. All senders are also receivers and there are 48 members for the 96 node network, 64 members for the 128 node network, and 96 members for the 192 node network. The result is that in the 192 node, node degree 8.4 case, we see in Figure 7.19 (b) that we have one pair that incurs almost 90% of the maximum penalty.

The cases of individual receivers with a penalty is evident in Figure 7.16 where there are several clear light blue and green vertical line patterns which indicate that receivers in the different clusters incur the receiving cluster penalty for almost all senders. The same is true of some senders with yellow horizontal line patterns in Figure 7.14.

The larger node degree does not have an impact in the DIS case as there are many members all over the clusters giving an increased likelihood of getting a good MBR over a large group. The connections between the members numbered 50 through 60 do not have a good connection with the members numbered 40 through 50 in Figure 7.18(c). These members incur a high penalty due to the likelihood of getting an MBR or a DBR far away in a two large cluster case.

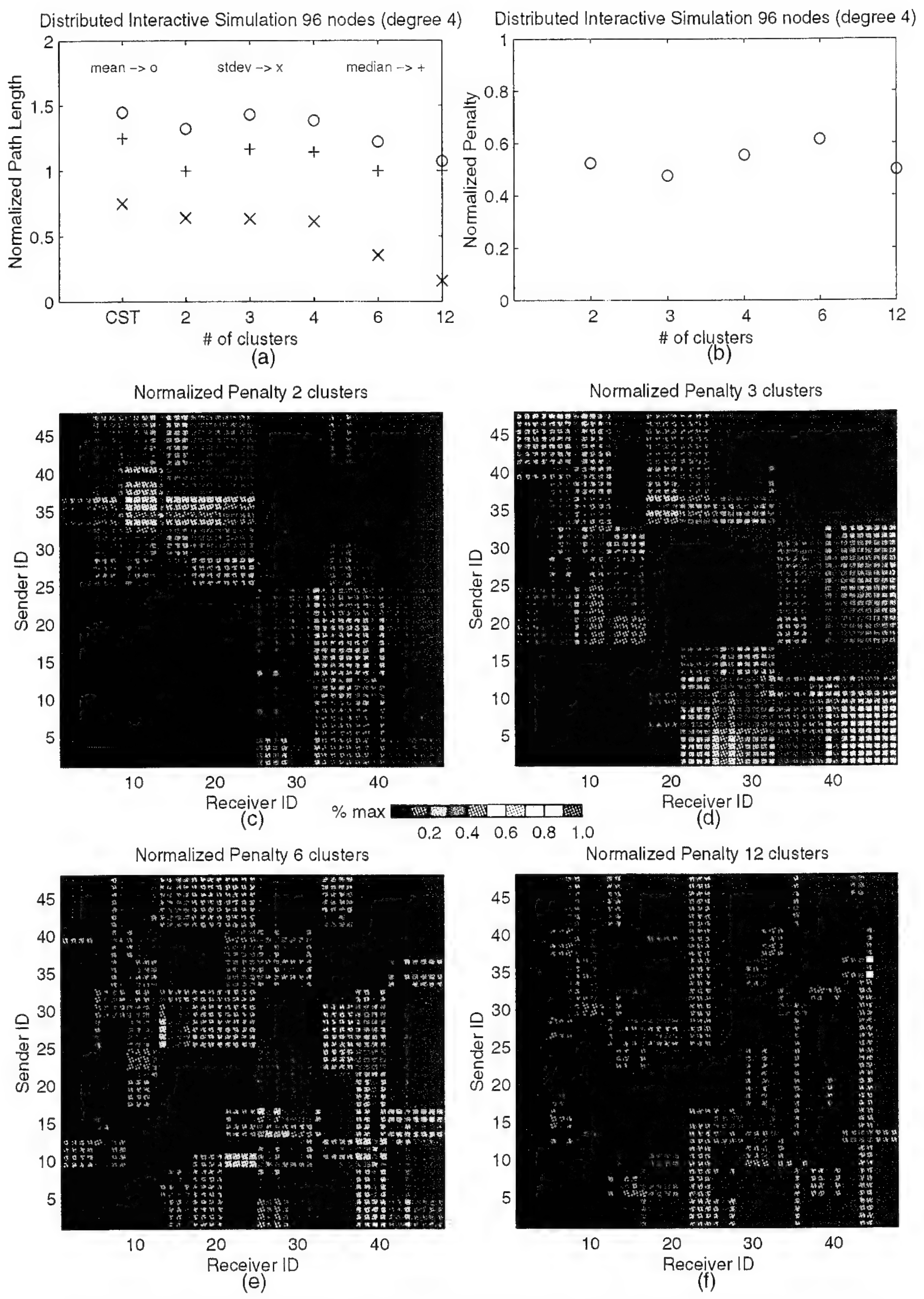


Figure 7.14 Distributed Interactive Simulation 96 Node Case Node Degree 4

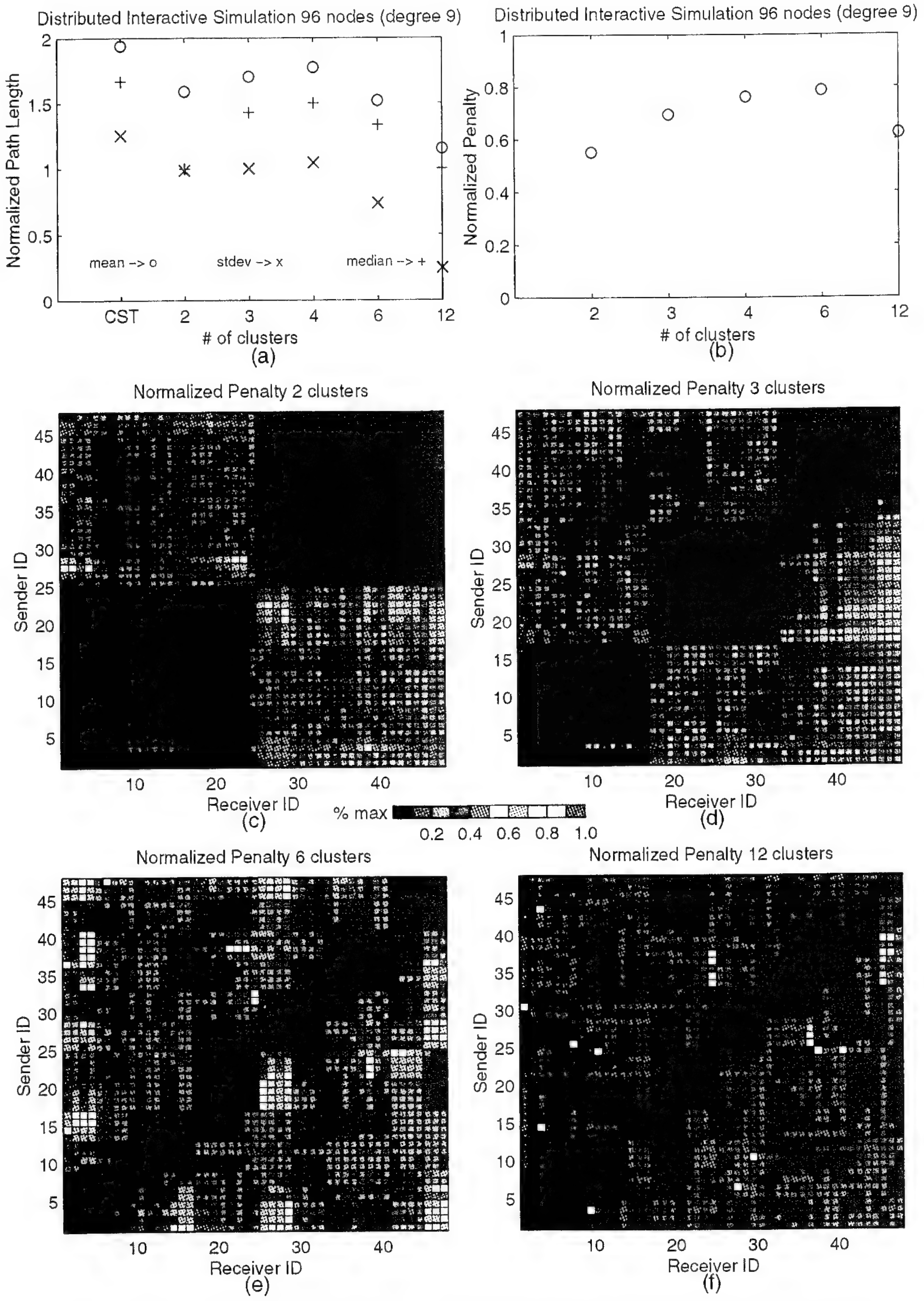


Figure 7.15 Distributed Interactive Simulation 96 Node Case Node Degree 9

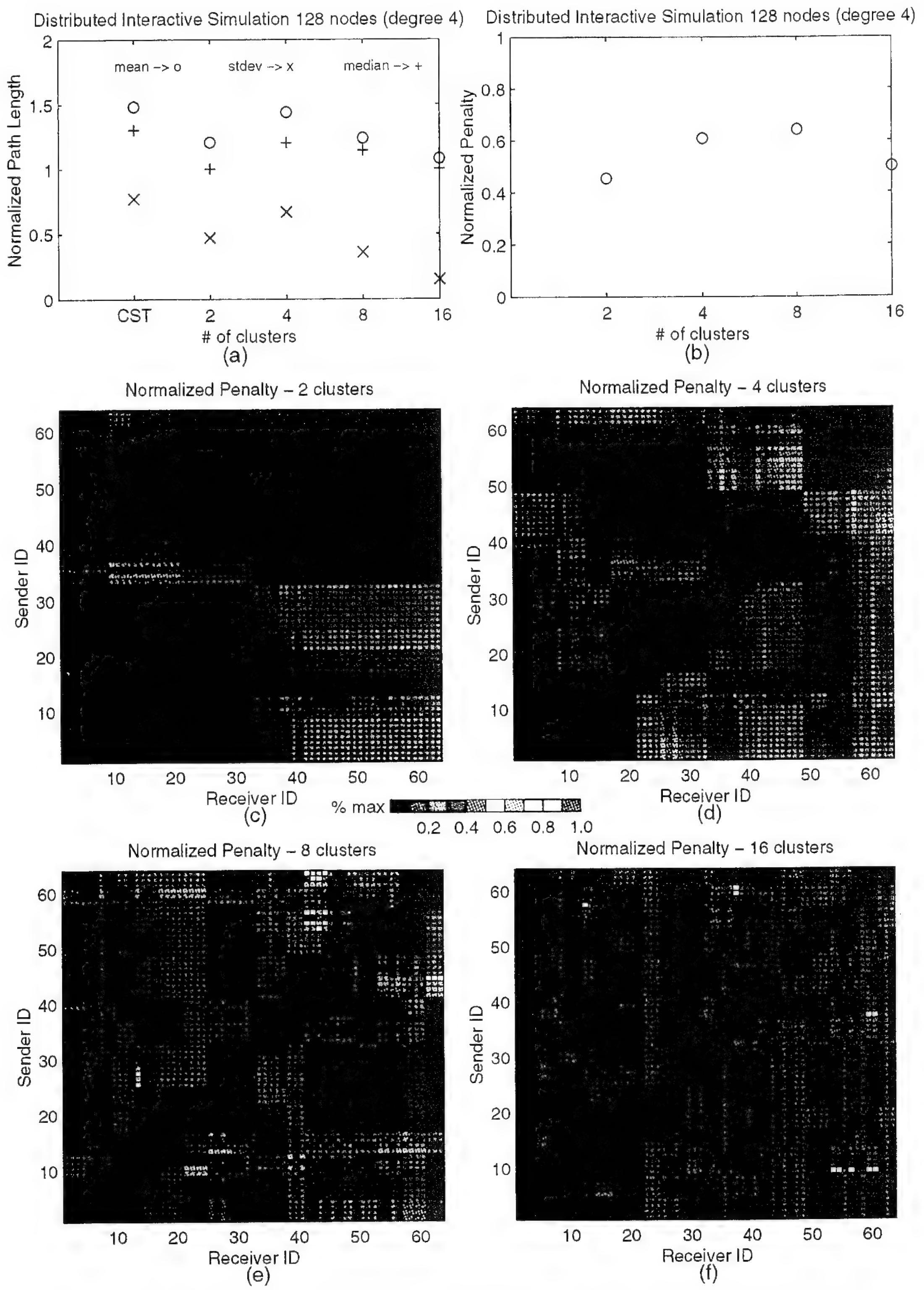


Figure 7.16 Distributed Interactive Simulation 128 Node Case Node Degree 4

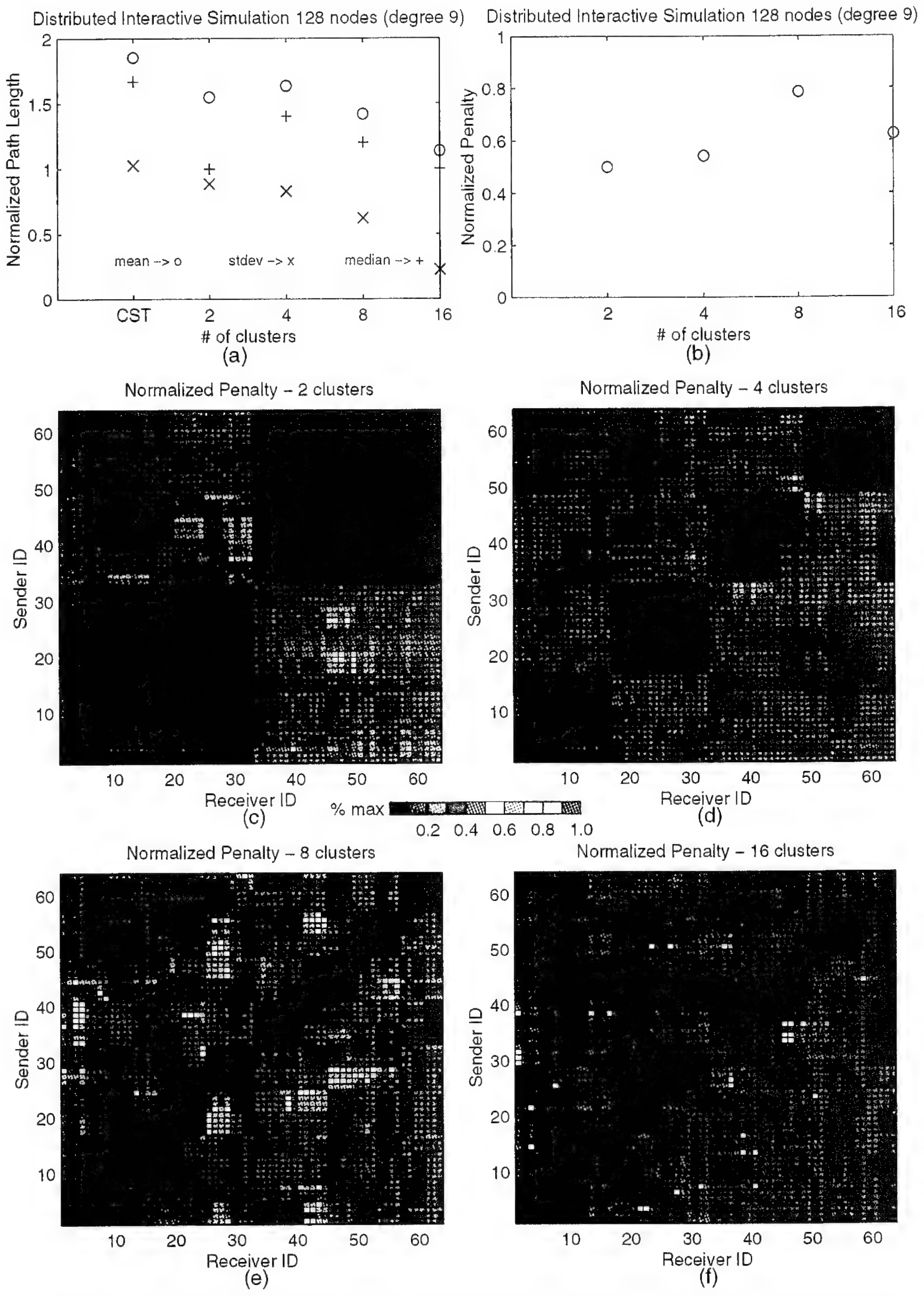


Figure 7.17 Distributed Interactive Simulation 128 Node Case Node Degree 9

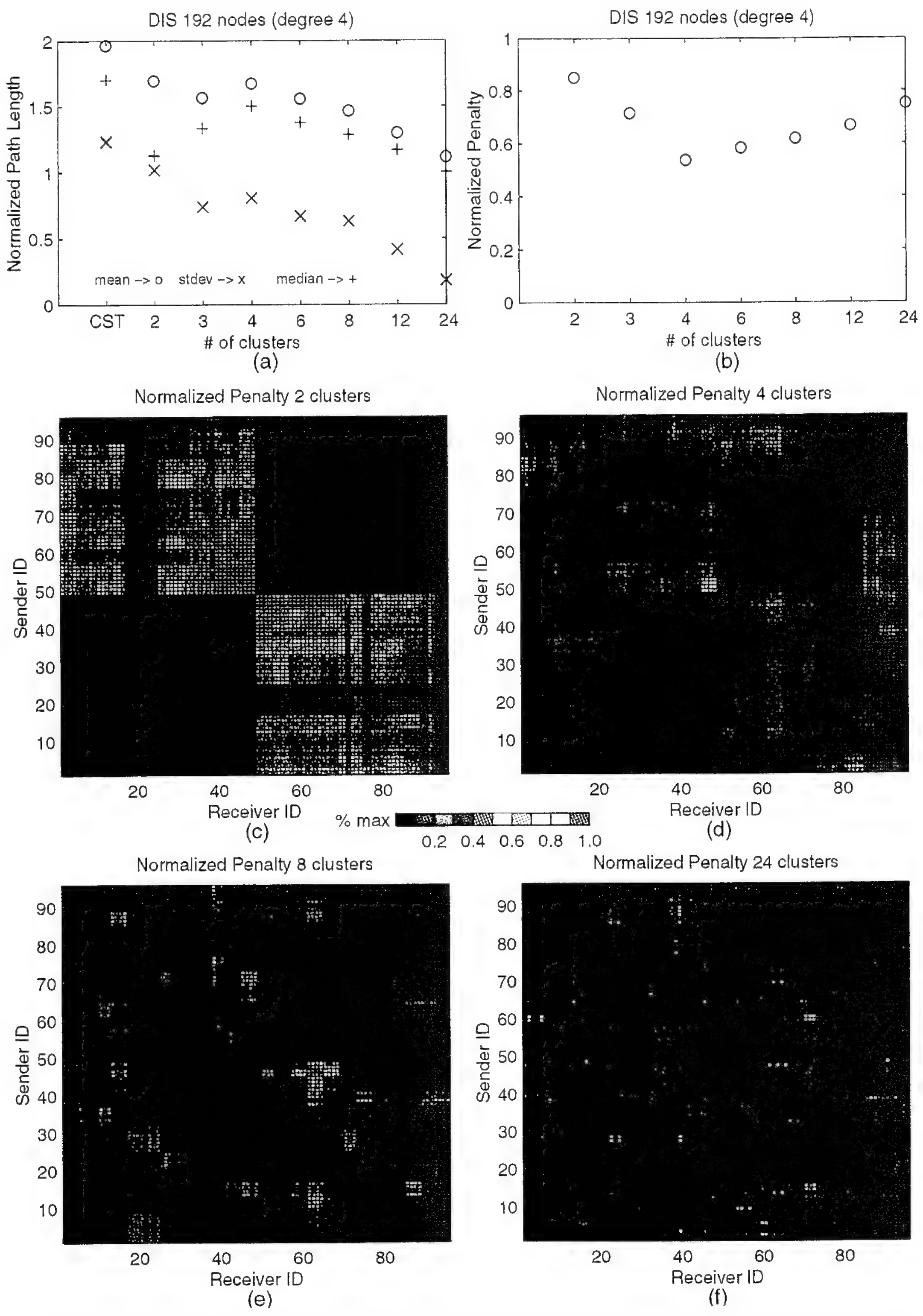


Figure 7.18 Distributed Interactive Simulation 192 Node Case Node Degree 4

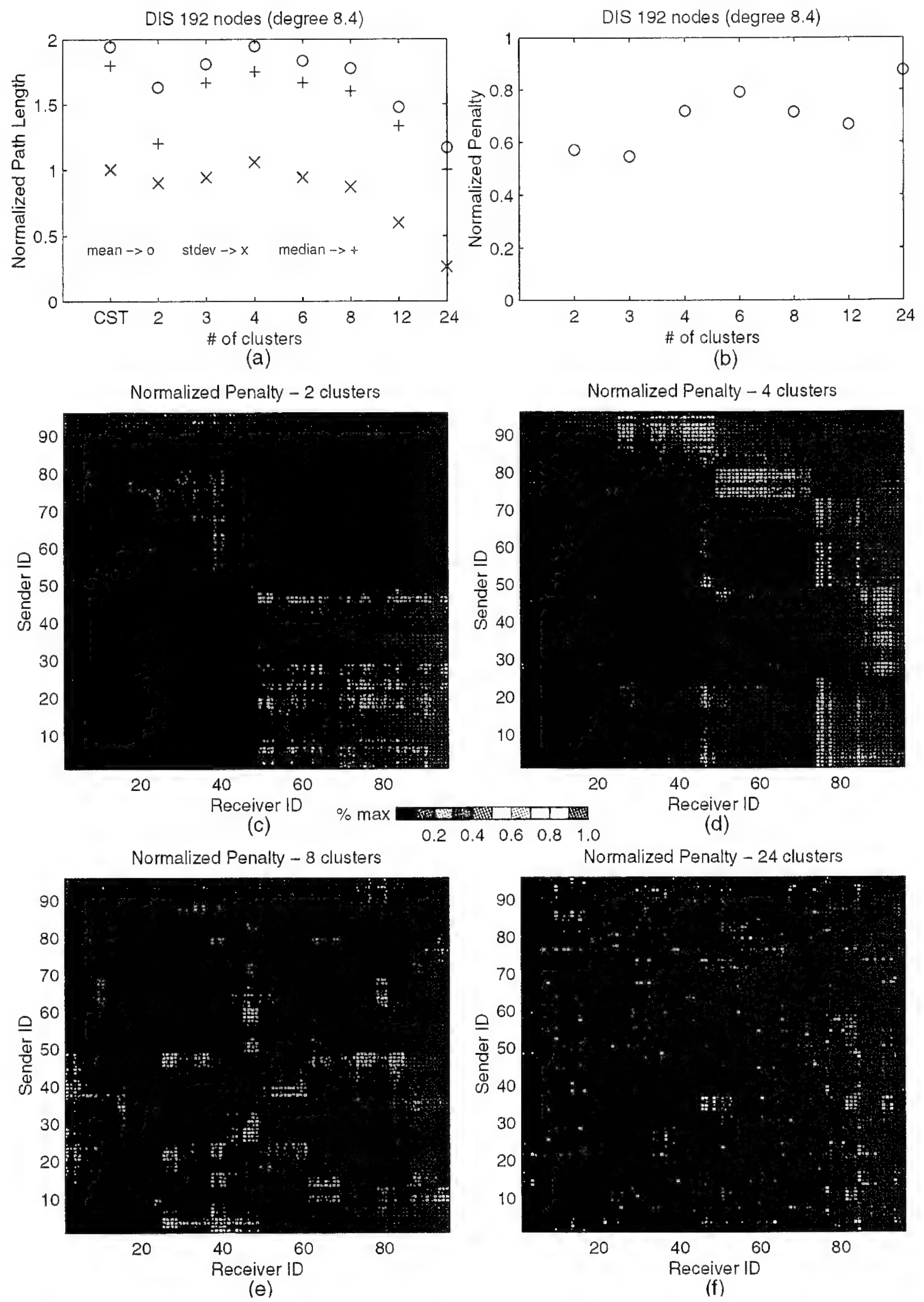


Figure 7.19 Distributed Interactive Simulation 192 Node Case Node Degree 8.4

E. SUMMARY OF RESULTS

The results from our simulations show that:

- The hierarchy, depending on the number of clusters chosen, gives path lengths approaching the shortest path.
- The worst case penalty bound is never exceeded.
- As the number of clusters goes up, a larger percentage of receivers incur a penalty, but the size of the penalty is smaller.
- For the random topologies generated, clustering does not reduce the diameter initially. This is reflected in the initial increase of path length in most cases.
- For randomly chosen centers, clustering performance is initially comparable to center-specific trees. It becomes superior to CSTs as the number of clusters goes up.
- When clustering is used, we force all paths inside of the cluster. Although we artificially impose this only for the clustered routing, we do not see this forced routing often. However, when we do force this, it results in a penalty inside the source cluster.

In the case of the electronic classroom, even though the center forms a shortest path tree to all receivers, clustering performance compares well.

For video conferencing, the results showed the sending and receiving cluster penalties clearly. The hierarchy performed well, approaching the shortest path in all cases as the cluster sizes got smaller.

For DIS, as we have many nodes participating, the number of nodes with a penalty increases. However, the penalty in all cases drops off substantially when there are more than 6 clusters. Even though we have more sender/receiver pairs with close to the worst case penalty, the number is still small relative to the size of the group.

In none of the simulations, is the worst case penalty reached. This shows that the likelihood of both a sender and receiver being in the worst case scenario is small. The more likely case is that both incur a penalty which is frequently much less than the worst case.

VIII. CONCLUDING REMARKS

A. CONCLUSIONS

The focus of the work presented in this dissertation is a new cluster-based hierarchical architecture for multicasting in internetworks and the investigation of its performance in terms of the path length metric. The assumptions behind this work are that all routers in the network are multicast-capable, an underlying two level hierarchy of domains and border routers exists, and that these border routers either have or can get the distance to each other. The proven and field-tested concept of a receiver-initiated join is maintained in the hierarchy.

The primary motivation behind the development of this architecture is the lack of scope control and scalability features in existing techniques. We have deployed a hierarchical organization of components such as the registrars and multicast border routers in a manner that maintains physical routing of multicast data along the shortest path between the leaf level clusters of the hierarchy.

We summarize the essential features of the hierarchical multicast architecture described in this thesis as below.

- *Scope Control:* The logical hierarchy controls the flow of multicast data through the use of clustering and border routers without sacrificing the benefits of a flat multicast address. In controlling the scope of the traffic, the network overhead is reduced by restricting the traffic from going to destinations where it is not required to go.

- *Performance:* The worst case path length penalty is bounded by the sum of the sending and receiving cluster diameters. Given hierarchical unicast and collection of a measure of goodness by the border routers, the worst case penalty can be reduced to the diameter of the leaf level clusters.

- *Operation:* Senders form shortest path trees to all receivers in their own source cluster. Receivers, with external sources, join towards their nearest border router at the level of the group. Cluster border routers, in receiving clusters, choose between themselves which one will act as the designated border router for a source cluster. Designated border routers join back to the source cluster forming a source-based shortest path tree. Border routers with receivers attached join towards the designated border routers. Border routers in a source cluster who have been chosen as the source cluster point of attachment for a designated border router, join the actual senders in their cluster.

- *Scalability:* This architecture provides for sender aggregation as designated border routers are chosen on a per source cluster basis, rather than per sender or group basis. It eliminates the need to locate centers at run-time either as destinations to join towards or collect sender information from. This is done by configuring a hierarchy into the network administratively.

We have provided an architecture that is capable of deploying protocols such as PIM or CBT in each leaf level cluster. By organizing the information about the hierarchy and presence of senders with registrars that are also organized hierarchically, we separate the resource discovery aspect of multicast completely from the tree construction aspect.

B. FUTURE WORK

We have shown that the proposed hierarchy performs well with respect to the path length and has other useful properties. Now, several aspects of multicasting in datagram internetworks in general, and CHARM in particular, can be pursued further.

The operation of the different components needs to specified in formal terms using a specification tools such as Lundy's in Reference [34] to make the interaction and synchronization between the different components concrete. This will enable identification of the synchronization requirements in situations such as DBR hand-off, DBR election, level *n* to leaf level registrar exchanges, etc. Such a specification is the logical next step towards an implementation of this architecture via a set of protocols. Similar work is also required for the architecture of the registrar hierarchy outlined here.

The schemes proposed for improving the worst case penalties by deploying a unicast hierarchy and introducing a measure of goodness collection by multicast border routers need to be evaluated in terms of their impact on the path length performance. A suitable measure for the goodness of a border router with respect to its internal senders needs to be identified and evaluated.

By choosing the clusters in a source-specific manner, we have performed aggregation of senders in a cluster. We note that the inter-cluster trees are built rooted at the source cluster. This potentially permits identification of clusters that have a common set of groups in operation. The traffic for these groups can be aggregated on the inter-cluster links to further save the state maintained by cluster border routers. This needs a mechanism for a cluster border router to join or leave an aggregated inter-cluster tree. Such a mechanism needs to be developed and evaluated for CHARM.

We have bound the scope of a group to a particular level of the hierarchy at group creation time. For dynamic groups, it is entirely possible that the anticipated level of the group changes during the group operation either due to all members in certain parts of the hierarchy leaving the interaction or due to completely new parts of the hierarchy attempting to join the interaction. In the present architecture, the latter scenario is not possible at all and the former scenario leads to the group remaining operational at a level higher than necessary. The ability to handle these situations requires addition of a mechanism for run-time scope level modification.

Finally, the performance of CHARM needs to be investigated with respect to group dynamics, traffic concentration, and control traffic bandwidth consumption using an analysis tool that generates various topologies, groups, and member behaviors.

APPENDIX A. SAMPLE INPUT FILE USED FOR CHARM SIMULATIONS

The following file is a sample input file for the 128 node video conference scenario, with node degree 4. The internetwork is defined as a series of domain adjacency tables. These are then connected using the Border Group Interdomain Adjacency Table. The number of links in this table determines the inter-domain node degree. Group information includes the number, location and expected role of the members. The nodes are identified using a 2 place decimal notation signifying their domain and their ID in that domain.

The hierarchy information in the input file starts with the tag "%%Internetwork Mulitcast Hierarchy%%." The input file shown is set up to run four different topologies each with a different clustering. The hierarchy has 3 levels, the domains are at level 1, the clusters of domains are at level 2 and the entire network is at level 3. The "%%Clusters%%" tag indicates the number of clusters in each level above 1. The number of domains is the number of level 1 clusters. The "%%Parent Cluster Definitions%%" tag indicates to the simulator how the level 1 to level *m*-1 clusters are to be clustered by assigning them to a parent cluster ID. All parent cluster IDs at level *m*-1 should be 1 to be in the same network.

The simulator reads in this file and simulates the proper level in the hierarchy based on where the members are.

```
//This file contains internetwork data for use with mast.exe

%%MAST Topology Data%%
%%Number of Domains%%
16

%%Domain Adjacency Data%%

%%Adjacency Table 1    %%
8
0 , 0 , 0 , 1 , 0 , 0 , 0 , 0
0 , 0 , 0 , 0 , 1 , 1 , 1 , 1
0 , 0 , 0 , 0 , 1 , 1 , 0 , 1
1 , 0 , 0 , 0 , 1 , 0 , 0 , 0
0 , 1 , 1 , 1 , 0 , 0 , 1 , 1
0 , 1 , 1 , 0 , 0 , 0 , 1 , 0
```

```
0 , 1 , 0 , 0 , 1 , 1 , 0 , 1
0 , 1 , 1 , 0 , 1 , 0 , 1 , 0

%%Adjacency Table 2    %%
8
0 , 0 , 1 , 0 , 1 , 0 , 0 , 1
0 , 0 , 0 , 1 , 1 , 0 , 1 , 1
1 , 0 , 0 , 1 , 1 , 0 , 1 , 0
0 , 1 , 1 , 0 , 0 , 1 , 1 , 1
1 , 1 , 1 , 0 , 0 , 0 , 0 , 0
0 , 0 , 0 , 1 , 0 , 0 , 0 , 0
0 , 1 , 1 , 1 , 0 , 0 , 0 , 0
1 , 1 , 0 , 1 , 0 , 0 , 0 , 0

%%Adjacency Table 3    %%
8
0 , 1 , 0 , 1 , 0 , 0 , 0 , 1
1 , 0 , 1 , 0 , 0 , 0 , 1 , 1
0 , 1 , 0 , 0 , 0 , 0 , 0 , 0
1 , 0 , 0 , 0 , 0 , 1 , 1 , 0
0 , 0 , 0 , 0 , 0 , 1 , 0 , 0
0 , 0 , 0 , 1 , 1 , 0 , 1 , 0
0 , 1 , 0 , 1 , 0 , 1 , 0 , 1
1 , 1 , 0 , 0 , 0 , 0 , 1 , 0

%%Adjacency Table 4    %%
8
0 , 0 , 0 , 0 , 1 , 0 , 1 , 0
0 , 0 , 1 , 1 , 0 , 0 , 0 , 1
0 , 1 , 0 , 1 , 0 , 0 , 0 , 0
0 , 1 , 1 , 0 , 1 , 1 , 1 , 0
1 , 0 , 0 , 1 , 0 , 1 , 0 , 1
0 , 0 , 0 , 1 , 1 , 0 , 0 , 0
1 , 0 , 0 , 1 , 0 , 0 , 0 , 0
0 , 1 , 0 , 0 , 1 , 0 , 0 , 0

%%Adjacency Table 5    %%
8
0 , 0 , 0 , 1 , 0 , 0 , 0 , 1
0 , 0 , 1 , 0 , 0 , 0 , 1 , 0
0 , 1 , 0 , 1 , 1 , 1 , 0 , 0
1 , 0 , 1 , 0 , 1 , 1 , 0 , 0
0 , 0 , 1 , 1 , 0 , 1 , 0 , 0
0 , 0 , 1 , 1 , 1 , 0 , 1 , 0
0 , 1 , 0 , 0 , 0 , 1 , 0 , 0
1 , 0 , 0 , 0 , 0 , 0 , 0 , 0

%%Adjacency Table 6    %%
8
0 , 0 , 1 , 0 , 1 , 0 , 0 , 0
0 , 0 , 1 , 0 , 1 , 0 , 1 , 0
```

```
1 , 1 , 0 , 1 , 1 , 0 , 1 , 0
0 , 0 , 1 , 0 , 1 , 0 , 0 , 0
1 , 1 , 1 , 1 , 0 , 1 , 1 , 1
0 , 0 , 0 , 0 , 1 , 0 , 0 , 0
0 , 1 , 1 , 0 , 1 , 0 , 0 , 0
0 , 0 , 0 , 0 , 1 , 0 , 0 , 0

%%Adjacency Table 7    %%
8
0 , 1 , 1 , 1 , 1 , 0 , 1 , 0
1 , 0 , 1 , 0 , 1 , 0 , 0 , 0
1 , 1 , 0 , 0 , 0 , 1 , 0 , 0
1 , 0 , 0 , 0 , 0 , 0 , 0 , 1
1 , 1 , 0 , 0 , 0 , 0 , 1 , 0
0 , 0 , 1 , 0 , 0 , 0 , 0 , 1
1 , 0 , 0 , 0 , 1 , 0 , 0 , 0
0 , 0 , 0 , 1 , 0 , 1 , 0 , 0

%%Adjacency Table 8    %%
8
0 , 1 , 1 , 0 , 1 , 1 , 1 , 0
1 , 0 , 1 , 1 , 0 , 0 , 1 , 0
1 , 1 , 0 , 1 , 1 , 0 , 0 , 0
0 , 1 , 1 , 0 , 1 , 0 , 0 , 0
1 , 0 , 1 , 1 , 0 , 0 , 0 , 1
1 , 0 , 0 , 0 , 0 , 0 , 1 , 1
1 , 1 , 0 , 0 , 0 , 1 , 0 , 0
0 , 0 , 0 , 0 , 1 , 1 , 0 , 0

%%Adjacency Table 9    %%
8
0 , 1 , 1 , 1 , 1 , 1 , 0 , 0
1 , 0 , 0 , 0 , 1 , 1 , 0 , 0
1 , 0 , 0 , 0 , 0 , 0 , 0 , 0
1 , 0 , 0 , 0 , 0 , 0 , 0 , 1
1 , 1 , 0 , 0 , 0 , 0 , 0 , 1
1 , 1 , 0 , 0 , 0 , 0 , 0 , 0
0 , 0 , 0 , 0 , 0 , 0 , 0 , 1
0 , 0 , 0 , 1 , 1 , 0 , 1 , 0

%%Adjacency Table 10    %%
8
0 , 0 , 0 , 1 , 1 , 1 , 0 , 1
0 , 0 , 0 , 0 , 1 , 1 , 0 , 1
0 , 0 , 0 , 0 , 0 , 0 , 1 , 0
1 , 0 , 0 , 0 , 1 , 0 , 0 , 0
1 , 1 , 0 , 1 , 0 , 0 , 0 , 1
1 , 1 , 0 , 0 , 0 , 0 , 1 , 1
0 , 0 , 1 , 0 , 0 , 1 , 0 , 1
1 , 1 , 0 , 0 , 1 , 1 , 1 , 0
```

```
%%Adjacency Table 11    %%
8
0 , 1 , 1 , 1 , 0 , 0 , 0 , 0
1 , 0 , 0 , 1 , 0 , 1 , 1 , 1
1 , 0 , 0 , 0 , 1 , 1 , 1 , 0
1 , 1 , 0 , 0 , 1 , 0 , 1 , 1
0 , 0 , 1 , 1 , 0 , 1 , 1 , 0
0 , 1 , 1 , 0 , 1 , 0 , 1 , 0
0 , 1 , 1 , 1 , 1 , 1 , 0 , 0
0 , 1 , 0 , 1 , 0 , 0 , 0 , 0

%%Adjacency Table 12    %%
8
0 , 1 , 1 , 0 , 1 , 1 , 0 , 1
1 , 0 , 0 , 1 , 0 , 1 , 0 , 0
1 , 0 , 0 , 0 , 1 , 0 , 0 , 0
0 , 1 , 0 , 0 , 1 , 0 , 1 , 1
1 , 0 , 1 , 1 , 0 , 1 , 1 , 0
1 , 1 , 0 , 0 , 1 , 0 , 0 , 0
0 , 0 , 0 , 1 , 1 , 0 , 0 , 0
1 , 0 , 0 , 1 , 0 , 0 , 0 , 0

%%Adjacency Table 13    %%
8
0 , 1 , 0 , 0 , 1 , 0 , 0 , 1
1 , 0 , 0 , 1 , 0 , 0 , 0 , 0
0 , 0 , 0 , 0 , 0 , 0 , 0 , 1
0 , 1 , 0 , 0 , 1 , 1 , 1 , 0
1 , 0 , 0 , 1 , 0 , 1 , 1 , 0
0 , 0 , 0 , 1 , 1 , 0 , 1 , 1
0 , 0 , 0 , 1 , 1 , 1 , 0 , 0
1 , 0 , 1 , 0 , 0 , 1 , 0 , 0

%%Adjacency Table 14    %%
8
0 , 0 , 0 , 1 , 0 , 0 , 1 , 0
0 , 0 , 0 , 1 , 0 , 0 , 1 , 0
0 , 0 , 0 , 0 , 0 , 0 , 1 , 0
1 , 1 , 0 , 0 , 1 , 0 , 1 , 1
0 , 0 , 0 , 1 , 0 , 0 , 1 , 1
0 , 0 , 0 , 0 , 0 , 0 , 0 , 1
1 , 1 , 1 , 1 , 1 , 0 , 0 , 0
0 , 0 , 0 , 1 , 1 , 1 , 0 , 0

%%Adjacency Table 15    %%
8
0 , 0 , 0 , 1 , 1 , 0 , 0 , 0
0 , 0 , 1 , 0 , 0 , 1 , 0 , 0
0 , 1 , 0 , 0 , 1 , 0 , 1 , 0
1 , 0 , 0 , 0 , 1 , 0 , 0 , 0
1 , 0 , 1 , 1 , 0 , 0 , 1 , 1
```

```
0 , 1 , 0 , 0 , 0 , 0 , 1 , 1
0 , 0 , 1 , 0 , 1 , 1 , 0 , 1
0 , 0 , 0 , 0 , 1 , 1 , 1 , 0

%%Adjacency Table 16    %%
8
0 , 1 , 0 , 1 , 1 , 0 , 1 , 0
1 , 0 , 0 , 1 , 0 , 0 , 0 , 1
0 , 0 , 0 , 1 , 0 , 0 , 0 , 1
1 , 1 , 1 , 0 , 1 , 1 , 1 , 1
1 , 0 , 0 , 1 , 0 , 0 , 0 , 1
0 , 0 , 0 , 1 , 0 , 0 , 1 , 0
1 , 0 , 0 , 1 , 0 , 1 , 0 , 0
0 , 1 , 1 , 1 , 1 , 0 , 0 , 0

%%Border Group Interdomain Adjacency Table%%
27
1 , 4 , 2 , 2 , 1 , 1
2 , 5 , 3 , 1 , 1 , 1
4 , 5 , 5 , 8 , 1 , 1
7 , 1 , 8 , 4 , 1 , 1
10 , 2 , 11 , 5 , 1 , 1
13 , 4 , 14 , 2 , 1 , 1
9 , 2 , 4 , 4 , 1 , 1
3 , 4 , 4 , 2 , 1 , 1
5 , 1 , 6 , 3 , 1 , 1
16 , 6 , 10 , 2 , 1 , 1
3 , 7 , 9 , 5 , 1 , 1
4 , 1 , 2 , 2 , 1 , 1
13 , 5 , 12 , 6 , 1 , 1
14 , 8 , 11 , 4 , 1 , 1
9 , 8 , 10 , 6 , 1 , 1
16 , 8 , 15 , 1 , 1 , 1
14 , 2 , 10 , 1 , 1 , 1
8 , 4 , 2 , 3 , 1 , 1
1 , 4 , 7 , 2 , 1 , 1
2 , 5 , 6 , 4 , 1 , 1
11 , 2 , 12 , 2 , 1 , 1
14 , 1 , 15 , 8 , 1 , 1
9 , 2 , 2 , 3 , 1 , 1
8 , 8 , 9 , 5 , 1 , 1
15 , 2 , 12 , 5 , 1 , 1
6 , 2 , 8 , 3 , 1 , 1
8 , 4 , 11 , 4 , 1 , 1

//This Network topology has
//a tolal calculated Node Degree of: 4.5

%%Multicast Groups%%
4
```

```
%%Group 1%%
%%Senders%%
24
1 , 2
1 , 5
2 , 3
3 , 7
4 , 4
5 , 2
5 , 5
6 , 1
6 , 5
7 , 4
8 , 2
8 , 7
9 , 1
9 , 6
10 , 5
11 , 2
11 , 7
12 , 1
12 , 8
13 , 2
13 , 3
15 , 1
15 , 6
16 , 4

%%Receivers%%
32
1 , 2
1 , 5
2 , 3
2 , 5
3 , 2
3 , 7
4 , 4
4 , 7
5 , 2
5 , 5
6 , 1
6 , 5
7 , 3
7 , 4
8 , 2
8 , 7
9 , 1
9 , 6
10 , 3
10 , 5
11 , 2
```

```
11 , 7
12 , 1
12 , 8
13 , 2
13 , 3
14 , 7
14 , 8
15 , 1
15 , 6
16 , 4
16 , 8


%%Internetwork Mulitcast Hierarchy%%

%%Topology 1%%

%%Levels%%
3

%%Clusters%%
2
1

%%Parent Cluster Definitions%%
1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 2 , 2 , 2 , 2 , 2 , 2 , 2 , 2
1 , 1


%%Internetwork Mulitcast Hierarchy%%

%%Topology 2%%

%%Levels%%
3

%%Clusters%%
4
1

%%Parent Cluster Definitions%%
1 , 1 , 1 , 1 , 2 , 2 , 2 , 2 , 3 , 3 , 3 , 3 , 4 , 4 , 4 , 4
1 , 1 , 1 , 1

%%Internetwork Mulitcast Hierarchy%%

%%Topology 3%%

%%Levels%%
3
```

```
%%Clusters%%
8
1

%%Parent Cluster Definitions%%
1 , 1 , 2 , 2 , 3 , 3 , 4 , 4 , 5 , 5 , 6 , 6 , 7 , 7 , 8 , 8
1 , 1 , 1 , 1 , 1 , 1 , 1 , 1

%%Internetwork Mulitcast Hierarchy%%

%%Topology 4%%

%%Levels%%
3

%%Clusters%%
16
1

%%Parent Cluster Definitions%%
1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16
1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1
```

LIST OF REFERENCES

1. Anthony Ballardie, *A New Approach to Multicast Communications in a Datagram Internetwork*, Ph.D. Thesis, University College London, January 1995.

2. A. Ballardie, *Core Based Tree (CBT) Multicast, Architecture*, Internet Draft, draft-ietf-idmr-cbt-arch-03.txt, February 9, 1996.

3. Tony Ballardie, Paul Francis, and Jon Crowcroft, *Core Based Trees (CBT) An Architecture for Scalable Inter-Domain Multicast Routing,* in *ACM SIGCOMM*, September 1993.

4. R. Barton, E. Klinker, R. Voigt, S. Shukla, *MAST: A Tool for Analysis of Multicast Tree Quality*, Technical Report NPS-EC-94-017, Naval Postgraduate School, December 1994.

5. Dimitri Bertsekas and Robert Gallager, *Data Networks,* Prentice Hall, 1992

6. Frank Boesch, *Introduction to Basic Network Problems,* Proceedings of Symposia in Applied Mathematics, Volume 26, 1982.

7. Eric Boyer, *Multicast Communication With Guaranteed Quality of Service*, Master's Thesis, Naval Postgraduate School, December 1993.

8. R. Braudes, S. Zabele, *Requirements for Multicast Protocols*, Technical Report RFC 1458, Internet Engineering Task Force, Network Working Group, May 1993.

9. Steve Casner, *Frequently Asked Questions (FAQ) on the Multicast Backbone (MBONE)* available from ftp.isi.edu:mbone/faq.txt, December 22, 1994.

10. D. R. Cheriton and S. E. Deering, *Host Groups: A Multicast Extension for Datagram Internetworks*, ACM/IEEE Proceedings of the 9th Data Communications Symposium, pp. 172-179, September 1985.

11. Yogen Kantilal Dalal, *Broadcast Protocols in Packet Switched Computer Networks*, Ph.D. Thesis, Stanford University, April 1977.

12. Yogen K. Dalal and Robert M. Metcalfe, *Reverse Path Forwarding of Broadcast Packets,* Communications of the ACM, Vol. 21 No, 12, December 1978.

13. Stephen Deering, *Multicast Routing in a Datagram Network*, Ph.D. Thesis, Stanford University, December 1991.

14. Stephen E. Deering, *Host Extensions for IP Multicasting*, Technical Report RFC 1112, Internet Engineering Task Force, Network Working Group, August 1989.

15. Stephen Deering and David Cheriton, *Multicast Routing in Datagram Internetworks and Extended LANs*, ACM Transactions on Computer Systems, May 1990.

16. Stephen Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, and Ching-Gung Liu, *Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification*, Internet Draft, draft-ietf-idmr-pim-sm-spec-02.ps, September 7, 1995.

17. Stephen Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, and Liming Wei, *Protocol Independent Multicast-Dense Mode (PIM-DM): Protocol Specification*, Internet Draft, draft-ietf-idmr-pim-dm-spec-01.txt, January 19, 1996.

18. Stephen Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, and Liming Wei, *An Architecture for Wide-Area Multicast Routing*, in *ACM SIGCOMM*, August 1994.

19. Matthew Doar and Ian Leslie, *How Bad is Naive Multicast Routing?* Infocomm 1993.

20. W. Fenner, *Internet Group Management Protocol, Version 2*, Internet Draft, draft-ietf-idmr-igmp-v2-01.txt, September 25, 1995.

21. Kamoun Farouk, *Design Considerations for Large Computer Networks*, Ph.D. Thesis, University of California, Los Angeles, 1976.

22. J.G. Gallager, P.A. Humblet, and P.M. Spira, *A Distributed Algorithm for Minimum Weight Spanning Trees*, ACM Transactions on Programming Languages and Systems, Vol.5, No. 1, January 1983.

23. Alan Gibbons, *Algorithmic Graph Theory,* Cambridge University Press, 1985.

24. Mark Handley, Jon Crowcroft and Ian Wakeman, *Hierarchical Protocol Independent Multicast (HPIM)*, draft, November 11, 1995.

25. Mark Handley and Van Jacobson, *Session Description Protocol (SDP)*, Internet Draft, draft-ietf-mmusic-sdp-01.ps, November 22, 1995.

26. C. Hedrick, *Routing Information Protocol*, Technical Report RFC 1058, Internet Engineering Task Force, Network Working Group, June 1988.

27. F.K. Hwang, D.S. Richards, P. Winter, *The Steiner Tree Problem,* North-Holland, 1992.

28. IEEE Standards for Local Area Networks, *Token Ring Access Method and Physical Layer Specifications*, IEEE Std 802.5-1989, June 2, 1989.

29. ISO/IEEE/ANSI Information Processing Systems – Local Area Networks, *Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, ANSI/IEEE Std 802.3-1988 (ISO 8802-3:1989), 1989.

30. Van Jacobson, *The Session Directory Tool,* Lawrence Berkeley Laboratory, 1992.

31. Richard M. Karp, *The Computational Complexity of Network Problems,* Proceedings of Symposia in Applied Mathematics, Volume 26, 1982.

32. Leonard Kleinrock and Farouk Kamoun, Hierarchical Routing for Large Networks: Performance Evaluation and Optimization, Computer Networks, Vol. 1, pages 155-174, 1977.

33. Vachaspathi P. Kompella, Joseph C. Pasquale and George C. Polyzos, *Multicast Routing for Multimedia Communications,* IEEE/ACM Transactions on Networking, Vol.1, No. 3, June 1993.

34. G.M. Lundy and R.E. Miller, *Specification and Analysis of a Data Transfer Protocol using Systems of Communicating Machines*, Distributed Computing, Vol. 5, No.3, pp. 145-157, December 1991.

35. L. Kou, G. Markowsky, and L. Berman, *A Fast Algorithm for Steiner Trees,* Acta Informatica 15, pp. 141-145, 1981.

36. Michael R. Macedonia, David R. Pratt, Michael J. Zyda, *A Network Architecture for Large Scale Virtual Environments*, Working Draft, April 1994.

37. G. Malkin, *Routing Information Protocol Version 2*, RFC 1723, Internet Engineering Task Force, Network Working Group, November 1994.

38. M.D. Mesarovic, D. Macko and Y. Takahara, *Theory of Hierarchical, Multilevel Systems,* Academic Press Inc., 1970.

39. J. Moy, *OSPF Version 2*, Technical Report RFC 1583, Internet Engineering Task Force, Network Working Group, March 1994.

40. J. Moy, *Multicast Extensions to OSPF*, Technical Report RFC 1584, Internet Engineering Task Force, Network Working Group, March 1994.

41. Radia Perlman, *Interconnections,* Addison-Wesley, 1992.

42. Parameswaran Ramanthan and Kang G. Shin, *Delivery of Time-Critical Messages Using a Multiple Copy Approach*, ACM Transactions on Computer Systems, Vol 10, No. 2, May 1992.

43. Y. Rekhter and P. Gross, *Application of the Border Gateway Protocol in the Internet*, RFC 1655, Internet, Network Working Group, July 1994.

44. Y. Rekhter and T. Li, *A Border Gateway Protocol 4 (BGP-4)*, Internet Draft, draft-ietf-idr-bgp4-02.txt, January 1996.

45. Yakov Rekhter and Paul Traina, *IDRP for IP v4 and v6*, Internet Draft, draft-ietf-idr-idrp-v4v6-02.txt, January 25 1996.

46. Shridhar B. Shukla, J. Eric Klinker and Eric B. Boyer, *Multicast Tree Construction in Network Topologies with Asymmetric Link Loads,* NPS Technical Report # NPS-EC-94-012, September 30, 1994

47. Shridhar Shukla, J. Eric Klinker, and Robert Voigt, *A Protocol for Locating Multicast Data Distribution Centers Using Participant Registration,* NPS Technical Report # NPS-EC-95-005.

48. Ajit S. Thyagarajan, Stephen L. Casner and Stephen E. Deering, *Making the MBone Real*, Proceedings of INET 96, pp 465-473, Vol 1, June 1995.

49. Ajit S. Thyagarajan and Stephen E. Deering, *Hierarchical Distance Vector Multicast Routing for the MBone*, ACM Communications Architectures. Protocols and Applications Conference Proceedings, pages 60-66. ACM SIGCOMM, September 1995.

50. P. Traina, *BGP-4 Protocol Analysis*, RFC 1774, Internet, Network Working Group, March 1995.

51. Robert J. Voigt, Robert J. Barton and Shridhar B. Shukla, *A Tool for Configuring Multicast Data Distribution over Global Networks*, Proceedings of INET 96, pp 455-463, Vol 1, June 1995.

52. D. Waitzman, C. Partridge, and S. Deering, *Distance Vector Multicast Routing Protocol*, Technical Report RFC 1075, Internet, Network Working Group, November 1988.

53. David W. Wall, *Mechanisms for Broadcast and Selective Broadcast,* Technical Report 190, Stanford University, June 1980.

54. Bernard M. Waxman, *Routing of Multipoint Connections*, IEEE Selected Areas in Communications, December 1988.

55. Bernard M. Waxman, *Performance Evaluation of Multipoint Routing Algorithms*, IEEE INFOCOM 1993.

56. Liming Wei and Deborah Estrin, *A Comparison of Multicast Trees and Algorithms*, Draft Submitted to INFOCOM 1994, 1993.

57. Wei Yen and Ian F. Akyildiz, *A Hierarchical Multicast Routing Protocol,* Proceedings of IEEE High Performance Computer Communication Subsystem '95, August 1995.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0944,
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library, Code 013 2
   Naval Postgraduate School
   Monterey, CA 93943-5002

3. Chairman, Prof. Herschel H. Loomis, Jr., Code EC 2
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

4. Prof. Shridhar B. Shukla, Code EC/SH 3
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

5. Prof. Paul Moose, Code EC/Mo 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

6. Prof. Gilbert Lundy, Code CS/Ln 1
   Department of Computer Science
   Naval Postgraduate School
   Monterey, CA 93943-5121

7. Prof. Norman F. Schneidewind, Code SM/Ss 1
   Department of Systems Management
   Naval Postgraduate School
   Monterey, CA 93943-5121

8. LT Robert J. Barton, USN 1
   Program Management Office
   Strategic Systems Programs
   Code SPL-20A
   P.O. Box 3504
   Sunnyale, CA 94088-3504

9.	Prof. Craig Rasmussen, Code MA/Ra Department of Mathematics Naval Postgraduate School Monterey, CA 93943-5121	1
10.	LTC Khaled Shehata SGC 1837 Naval Postgradute School Monterey, CA 93943-1837	1
11.	Dr. Michael R. Macedonia Vice President Fraunhofer CRCG 167 Angell Street Providence, RI 02906	1
12.	CDR Robert J. Voigt USN c/o Special Projects Manager Bureau of Naval Personnel Sea Duty Component P.O. Box 16134 Arlington, VA 22215-1134	4
13.	Prof. Murali Tummala Code EC/Tu Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1